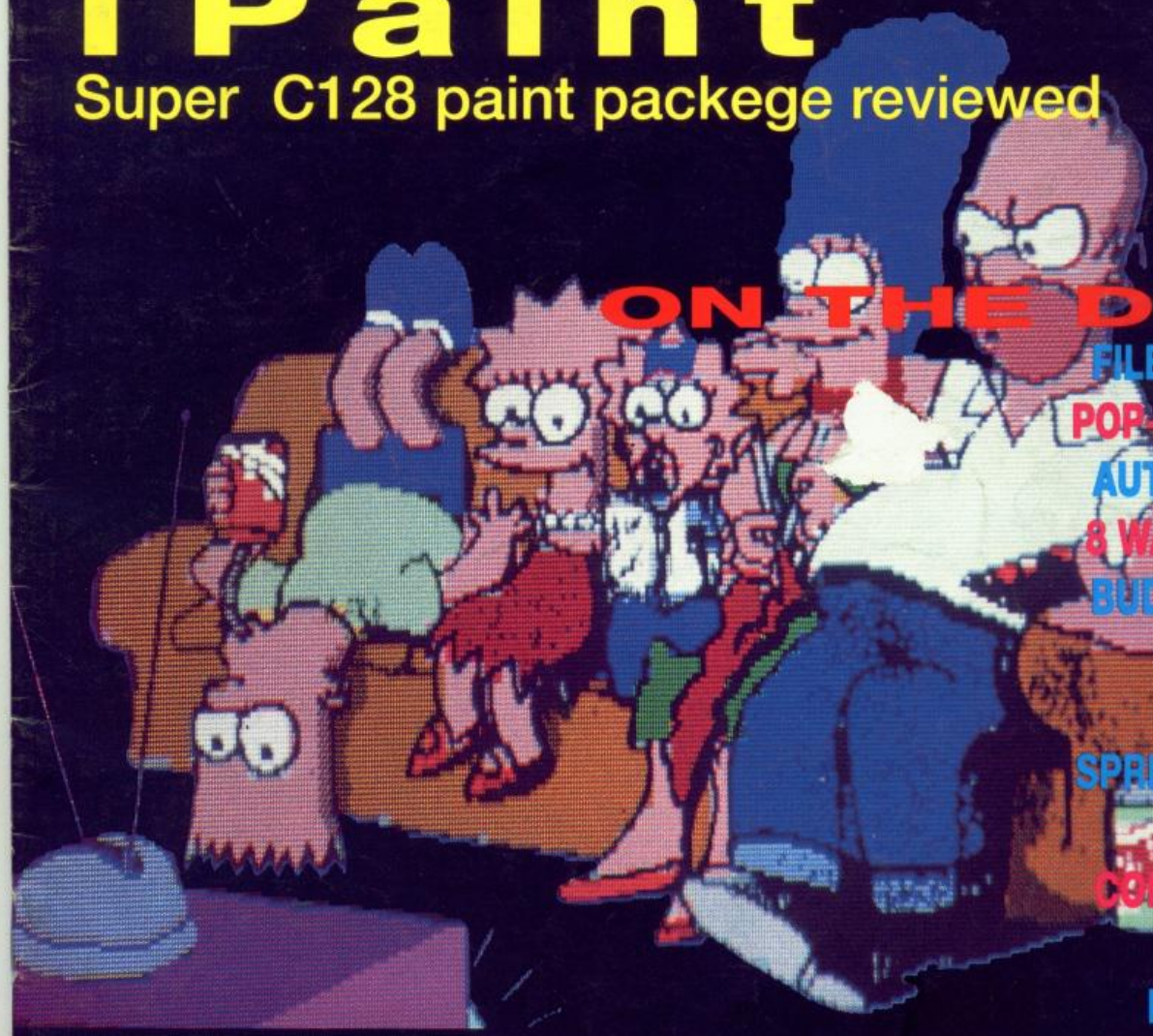
£3.25
DM.18

September 1991

# Commodore DISK·USER

# I Paint

## Super C128 paint packege reviewed



## ON THE DISK

FILE UTILITIES
POP-UP MENUS
AUTO SAVE 64
8 WAY SCROLL
BUDGET CALC
VERTICAL SCROLL
SPRITE EDITOR
UDG COMPRESSOR
SPRITE PRIORITIES
RETURNER
MOUSE 80(C128)

ISSN 0953-0614

09

9 770953 061014

# Elvira®

## mistress of the dark™

- Hour upon hour of addictive entertainment
- Totally icon-driven for ease of play
- Turbo disk loading guarantees fast graphical access
- An Award Winning game

**A SUPERB FANTASY ROLE-PLAYING GAME WITH OUTSTANDING GRAPHICS AND GAMEPLAY**

- Over 1 Megabyte of game code supplied on 3 double-sided disks
- Hundreds of locations in and around Elviras Haunted Castle
- Hand to hand real time combat with Knights and Monsters
- Includes instructions and secret spell book for mixing spells and potions

CASTLE RAMPARTS
CBM64

GARDENERS SHED
CBM64

CASTLE ARMOURY
CBM64

*AWESOME QUALITY: THE BEST FRP EVER SEEN ON THE CBM 64*

**ELVIRA** rescues you from the Jailers cell and sends you on an unforgettable mission to dispose of the spirit of her great great grandmother **EMELDA.** On your travels in and around the **HAUNTED CASTLE** you will come across some of the most weird and...

FALCON ATTACK
CBM64

....**EVIL** monsters and beings you have ever encountered. From **UNDEAD KNIGHTS** to the **CATACOMB BEAST, WEREWOLVES** to a **VAMPIRE,** and of course wicked **EMELDA** herself, this is the challenge that you will play **AGAIN AND AGAIN!**

ELVIRA IN THE KITCHEN
CBM64

A KNIGHT FIGHTING

ALL SCREENSHOTS FROM CBM64

THE BLACKSMITHS FORGE
CBM64

For information about the Elvira Fan Club:
14755 Ventura Blvd.,
#1-710, Sherman Oaks, CA 91403, USA

DESIGNED BY HORROR SOFT LTD

FLAIR SOFTWARE



**Available from your local dealer or direct from Flair Software Ltd, The Smithy Side, 7 Belle Villas, Ponteland, Newcastle Upon Tyne. NE20 9BD Priced £24.99**

# CONTENTS

Commodore DISK·USER·

Volume 4 Number 10 AUGUST 1991

## ON THE DISK



## IN THE MAGAZINE



**Group Editor:** Paul Eves
**Designer:** Mark Newton
**Technical Editor:** Jason Finch
**Program Evaluator:** John Simpson
**Publishing Consultant:** Paul Crowder
**Advertisement Manager:** Cass Gilroy
**Classified Sales:** Deborah Curran
**Publisher:** Hasnain Walji
**Distribution:** Seymour Press Distribution Ltd. Winsor House, 1270 London Road, Norbury, London SW16 4DH. Tel:081 679 1899. Fax: 081 679 8907
**Printed By:** Gibbons Barford Print

**Subscription Rates**

| | |
|---|---|
| UK | £33.00 |
| Europe | £39.00 |
| Middle East | £39.30 |
| Far East | £41.60 |
| Rest of World | £39.70 or $69.00 |

Airmail rates on request
Contact: Select Subscriptions. Tel: (0442) 876661

Commodore Disk User is a monthly magazine published on the 3rd Friday of every month. Alphavite Publications Limited, 20, Potters Lane, Kiln Farm, Milton Keynes, MK11 3HF. Telephone: (0908) 569819 FAX: (0908) 260229. For advertising ring (0908) 569819

# EDITORS COMMENT

WELCOME TO YET ANOTHER EDITION OF YOUR FAVOURITE C64 MAGAZINE. UNFORTUNATELY, THIS IS THE LAST ISSUE THAT YOURS TRULY WILL BE EDITING. AS FROM THE 1ST OF AUGUST I WILL HAVE GONE TO PASTURES NEW. I WOULD LIKE TO SAY THAT I HAVE THOROUGHLY ENJOYED WORKING ON THE MAGAZINE, AND HAVE APPRECIATED ALL YOUR PAST SUPPORT AND INTEREST. I SINCERELY HOPE THAT COMMODORE DISK USER CONTINUES TO THRIVE AND THAT YOU WILL ALL CONTINUE TO SUPPORT IT.

NOTE: THERE ARE A COUPLE OF ERRORS IN THE BASICS OF BASIC FEATURE THIS MONTH. IN THE SUBROUTINES SECTION YOU WILL NEED TO ADD A LINE IN THE RANDOM SECTION. LINE 2025 WHICH IS MISSING SHOULD READ, 2025 NEXT:GOSUB6000
SECONDLY, IN THE SORTING HIGH SCORES PROGRAM. LINE 3000 SHOULD READ ST(T) AND NT(T) NOT SD AND ND AS STATED.

IT JUST LEAVES ME TO SAY THANK YOU ALL AND ALL THE BEST FOR THE FUTURE.

## DISK INSTRUCTIONS

Although we do everything possible to ensure that CDU is compatible with all C64 and C128 computers, one point we must make clear is this. The use of 'Fast Loaders', 'Cartridges' or alternative operating systems such as 'Dolphin DOS', may not guarantee that your disk will function properly. If you experience problems and you have one of the above, then we suggest you disable them and use the computer under normal, standard conditions. Getting the programs up and running should not present you with any difficulties, simply put your disk in the drive and enter the command.

LOAD"MENU",8,1

Once the disk menu has loaded you will be able to start any of the programs simply by selecting the desired one from the list. It is possible for some programs to alter the computers memory so that you will not be able to LOAD programs from the menu correctly until you reset the machine. We therefore suggest that you turn your computer off and then on again, before loading each program.

## HOW TO COPY CDU FILES

You are welcome to make as many of your own copies of CDU programs as you want, as long as you do not pass them on to other people, or worse, sell them for profit. For people who want to make legitimate copies, we have provided a very simple machine code file copier. To use it, simply select the item FILE COPIER from the main menu. Instructions are presented on screen.

## DISK FAILURE

If for any reason the disk with your copy of CDU will not work on your system then please carefully re-read the operating instructions in the magazine. If you still experience problems then:

1. If you are a subscriber, return it to:
   Select Subscriptions Ltd
   5, River Park Estate
   Berkhamsted
   Herts
   HP4 1HL Telephone; 0442 876661

2. If you bought it from a newsagents, then return it to:

   CDU Replacements
   COPYTEC SOFTWARE SOLUTIONS
   Unit 29
   Riverside Business Centre
   Victoria Street
   High Wycombe
   Bucks
   HP11 2LT

Within eight weeks of publication date disks are replaced free.

After eight weeks a replacement disk can be supplied from COPYTEC SOFTWARE SOLUTIONS for a service charge of £1.00. Return the faulty disk with a cheque or postal order made out to COPYTEC SOFTWARE SOLUTIONS and clearly state the issue of CDU that you require. No documentation will be supplied.

Please use appropriate packaging, cardboard stiffener at least, when returning disk. Do not send back your magazine, only the disk please.

NOTE: Do not send your disks back to the above address if its a program that does not appear to work. Only if the DISK is faulty. Program faults should be sent to: BUG FINDERS, CDU, Alphavite Publications Ltd, Unit 20, Potters Lane, Kiln Farm, Milton Keynes, MK11 3HF. Thank you.

# I paint C128

**PAUL TRAYNOR checks out the latest paint package for the C128 and decides it's worth every penny**

**'Earths only Commodore 128 interlaced paint program' is proudly stated on the manuals cover, followed by copyright 1991, it is a pleasure to see that new products are still entering the Commodore 128 market. The next thing to be read on the manual cover is the list of required hardware: 128D/128 with 64k VDC RAM (using 80 column mode), 1351 or compatible two button mouse, properly connected RGBI or monochrome Video monitor, and a 1541/71 disk drive. If you own the above you are ready to experience something quite remarkable, a graphics package which really does match up to some 16 bit packages.**

## INITIALLY

The statistics are a remarkable 640x400 screen resolution with thousands of colours to paint with. In the past perhaps the best paint packages available have been used in 64 mode or 40 column 128 mode, now this package not only uses the 128's 80 column mode, but proves that this is potentially the more powerful and more suitable mode for graphics.

The package contents are a spiral bound manual and a program and files disk in addition to the I PAINT program there is a version of RAMDOS and a program for simply viewing I PAINT creations. The manual is A5 size, 40 pages long with an adequate index, 40 pages means it is long enough to explain every function but not too long to frighten off the user from reading it. The manual is split into a number of sections, one of which gives you quick start instructions for those users who are eager to see what I PAINT can do without having to spend time reading the whole of the text. You can be easily up and running within 30 minutes, your first action should be to make a backup copy of the disks. As well as creating backup disks for normal use you should format a further disk to hold all the printer drivers and fonts files. These drivers and fonts come as two compressed program files which should be run to create your standard drivers and font files automatically.

## RUNNING THE PACKAGE

When you run the program you can choose from a number of example picture files including Bugs Bunny, John Lennon and The Simpsons (who are of course the funniest family to come out of America since The Waltons). There are two disks the first contains all the program and example files it is a doubled sided disk to suit a 1571 drive if you use this disk on a 1541 drive you are not able to access some of the files, so a second disk is included which contains these files for the 1541 owners. Hence every one is catered for. Other appreciated sections in the the manual include a interesting one on printer drivers to aid you make the best

or most suitable selection and a helpful section on troubleshooting.
On booting you will get the choice of altering the setup file, this holds the defaults values for some of the functions. If you have a 1700 RAM expansion you can choose to use this as a RAM Disk, or an expansion of I PAINT's clip and safe areas. If you have a 1750 RAM expansion you can choose to use both options. You can choose left hand or right hand mouse, this swaps the functions of the two mouse buttons over, a left handed person may find this very useful. The following functions can both be altered in the setup file or when actually using the program: You may alter the ink and paper colours (i.e. foreground and background colours). The drawing options can also be altered these are referred to as: over, with (OR), thru (XOR) and mask (AND), these determine what happens when you draw or write on top of an existing object.

## MANY FEATURES

On booting it is not necessary to setup I PAINT, just select Start I PAINT and you will be presented with the title screen while the rest of the program loads. When loaded the main menu appears across the top of the screen. The menu is a row of icons. One nice feature here is that your pointer will only ever move within the area which you can choose options from at that time. i.e. at first you can move all across the top but if a sub-menu is selected pointer movement is then restricted to within it boundaries. All menus and sub-menus only ever occupy a narrow band at the top of the screen. This menu band disappears when you carrying out any drawing functions and you can also view the whole screen at any time with just the press of a mouse button. Choices available for drawing freehand are a pen, a spray can, or a range of brushes. If your freehand is not too good you can choose from a sub-menu of shapes which includes, line, triangle, rectangle, circle as well as a number of 3-D solids including spheres, toriods and spools. These 3-D shapes you may recognize from BASIC 8 if you are familiar with it. Selecting the palette icon from the main menu will allow you to choose or alter the screen colors. There are 4 colour areas which can be set these are ink; fields 1 and 2, and paper; fields 1 and 2. the standard 16 colors are shown in a bar for you to point and select for each field. Mixing of these standard colors will give you literally thousands of apparent colours e.g. setting ink colour field 1 to red and field 2 to yellow will give a orange shade when you draw or screen, this obviously multiplies up the number of colours available and this is further increased when you remember there are two fields of colour for the paper as well. A magnify mode is available for small detail work and this is one of the functions which benefits if you using an expansion RAM cartridge. If you do not have expansion RAM then you will lose any clip or safe area during magnification. This is no real problem as clips can always be stored on floppy disk for later retrieval. Selecting areas from the main menu gives quite a powerful set of functions, you can save an area as a clip which can be turned into a pattern as well as having an extra area of the screen set up as a safe area. If you have an expanded 128 you it is possible to have a safe area and a clip both full screen size along with a custom pattern about two thirds screen size. All in memory with a picture on screen as well. The safe area is designed to be used to temporarily save your picture perhaps while you make try a number of different alterations to your drawing knowing you can have it restored to the way it was when it was declared safe. A clip can be rotated, flipped or have its colours reversed and be pasted anywhere on screen or changed into a pattern. A pattern is brought back onto the screen using the fill command. It is possible for the fill command to loose it way on a large intricate or complicated area, for example one which contains 3-D solids, this should should not prove too much of a problem but is something to bear in mind. The Draw mode/pattern icon is used to select write, erase, pattern or custom pattern. There are 6 standard (or hard) patterns, a custom pattern can be stored to and loaded from disk. Font selection is handled in a similar way to the patterns, there are a number of standard (or hard) fonts along with a custom font which can be loaded from disk, I PAINT is compatible with BASIC 8 fonts and BASIC 8 patterns. Text can be put onto the screen in any of four directions and can also have its priority (over, with, thru and mask) and its orientation (flip, reflect, rotate and unrotate) controlled. Underlining, colour and reverse text can all be selected. Most of these functions for text mode are selected using CRTL, ALT, Function keys or the commodore logo key along with others. You can even set your own function key macros outside I PAINT which can then be used within I PAINT. Use of and creation of function key macros is

well explained in its section of the manual.

## BASIC 8 COMPATIBILITY

I PAINT is compatible with BASIC 8 pictures and brush files as well as Lacemaker pictures, although the colour information will not work properly for some of the color cell sizes. So when a BASIC 8 file is loaded it is often the best idea to colour wash the whole area (to clear the colours) and then use I PAINT's extended palette to re-paint. I PAINT picture files can be saved as compressed or un-compressed files and as you may have guessed can take up a lot of disk space, The example graphic files have an average size of about 70 blocks, these are all compressed files. I PAINT will support a range of printers using the included drivers as well as this it is able to to utilise, although not quite as efficiently in terms of speed, any BASIC 8 printer driver which you may have.

## INTERLACED SCREENS

A disadvantage of using interlace screens is that if you have two contrasting colours on fields 1 and 2 you will get flicker. Fine tuning of contrast and brightness controls on your monitor will help and I PAINT also gives you the option to work in monochrome mode, rather than colour, which may help. Some operations will also operate quicker whilst in monochrome mode. When you have a screen full of multi-colour graphics it is possible to lose sight of your pointer so there is an option to use cross hairs like the ones found on CAD programs which run the full length and width of the screen. I like this function very much as cross hairs are ideal for lining up graphic items on your screen. I would not recommend cross hairs for text though as I obtained some strange results.

I PAINT gets its high resolution screen from the fact that it can draw to both fields of the display screen as described, but the option is given to draw to just one field at a time, this can lead to some very interesting colour effects on screen. The last option across the top of the main menu is clear, this is a surprisingly powerful option, you can choose to clear a full screen or any defined window, you can choose from 9 different halftones to clear the screen or window with. These halftones and made up of varying degrees of the 2 ink and 2 paper colours, using these halftones for clearing windows further multiplies up the number of apparent screen colours available. You may also choose to do these clearing operations to only one of the two fields leaving the other intact.

## WHAT'S ESSENTIAL

When a software author produces his material he has to decide what items of hardware to have as essential and what items to make optional. The memory expander is optional. Any user who does not have a memory expansion will be able to access all the functions of I PAINT, but If you have spent good money on extra memory it is not wasted because of the ability of I PAINT to utilise it for both extra clip and safe areas as well as a RAM disk, using the version of RAMDOS supplied with the package. The RAMDrive can filled with files before using I PAINT and emptied of them after quitting. You can also choose to ignore the RAM and hence it could hold files from other programs while you are using I PAINT. I PAINT can be easily copied on to any drive type and booted from any device number so it is very suitable for use with a 1581 even within a partition or sub-directory if required.

There are 1 or 2 minor shortcomings, the micro function wasn't perfect and the manual would have benefitted from a picture file structure for the user who would want to program their own graphic conversion programs. I PAINT though is in a class of it's own as far as 128 graphics packages, the high degree of compatibility with various BASIC 8 files makes I PAINT easy to upgrade from BASIC 8 drawing packages, such as BASIC Paint, without having to recreate patterns, printer drivers and even whole pictures which you have created in the past. I PAINT would not look out of place if compared with 16 bit packages like Deluxe Paint 2. It makes excellent use of available peripherals and has a number of features which are not seen on any other packages. Make sure your machine possesses 64k VDC RAM, you have a proportional 1351 mouse and a RGBI or monochrome 80 column monitor before purchasing. Overall I PAINT represents value for money and is an important addition to any 128 users library.

**Program** I PAINT
**Supplier** F.S.S.L. Defford Road. Pershore. Worcs
Tel. (0386) 553153
**Price** 39.95

# BUDGET CALC

**Keep track of your money with this budget program - SIMON PHILLIPS**

Budget calc is a budget utility based around two spreadsheets. The first contains the actual monthly figures, the second contains the estimated figures. BUDGET CALC is split into several sections, the main program, review program, end program, changing subjects and changing months. The first three RUN automatically, and interact with each other. The last two programs have to be LOADed separately.

## CHANGING SUBJECTS AND MONTHS

BUDGET CALC has the capacity for 16 subjects, and these can be changed to suit the user. The subjects are stored in a sequential file on disk, the same for the month order. Each of the subjects must be no more than 12 characters in length.

For different people, financial years change. With BUDGET CALC, you can have your financial year from Jan to Dec.

When LOADing and RUNning BUDGET CALC, you will be asked for the present month, this is for the printout, file purpose, and the advance year facility. The program compares the month with the start of the financial year, and if they match, then the year is updated. Once the space bar is pressed, data is loaded. This is in three separate sequential files. The first data is the actual and estimated figures, along with back-dated figures, in the "budget subjects". The last is the month sequence.

Once the red light on the disk drive goes off, a command window is printed at the bottom of the screen, and the spreadsheet window is displayed in the upper and middle of the screen. The window is moved around the spreadsheet by using a joystick in pot (No 2). Pressing the fire-button and moving the joystick left or right will cause the large cursor to move through the five commands. The cursor has a wrap-round feature, and the inverse command denotes which mode you are in. Pressing the fire-button will execute these commands. They are - REVIEW, MOVE, AMEND, END, ACTUAL/ESTIMATED

## REVIEW

Further LOADing is required for this function and this is done automatically. Once LOADed, a menu screen is printed and five options are given:

**REVIEW MONTHS**
**REVIEW SUBJECTS**
**REVIEW TOTALS GRAPHICALLY**
**REVIEW PROFIT AND LOSS**
**RETURN TO THE MAIN PROGRAM**

The first two options allow the user to review figures back-dated by three years. Once these options are executed, only half of the months or subjects are displayed. By moving the joystick up and down, access can be gained to the rest of the figures. Pressing the fire button will display the menu once again. Review Totals displays graphs of the past and present annual totals. There are two sub-routines for this option. The first is for totals not exceeding 4000, and the second is for totals greater than 4000.

**Pressing the fire button will return you to the menu.**

Review profit displays the spreadsheet as a grid. After a couple of seconds boxes will be coloured either red or green. A red box means that at that position on the spreadsheet the actual figure is lower than the estimated. No doubt there will be a lot of red boxes at the end of the grid because there are as yet, no actual figures. Pressing the space bar will return you to the menu.

## MOVE

This command enables you to move around the spreadsheet.

## AMEND

This command is used in conjunction with the "move" command. It is used to change, add and correct figures. Use the move command to position the entry place on the screen. Enter the amend routine and press fire. Three reverse characters will be displayed in the top left hand corner of the first figure. This is now your cursor and you can move it around the four by four square of figures. Simply type in the new figure (only using three characters) and then press RETURN. If you have made a mistake in keying in the new value and you have not yet pressed RETURN , then type out the figure again and it will be printed over the mistake. If you have made a mistake and have pressed RETURN, then you will have to go through the routine again. If at any time you want to get out of the amend routine then press the fire-button and you will return to the move

command.

## END

After selecting this command, the "bud end" program will LOADed and RUN. The program does not need any explanation of how to use it. It allows the saving of data, and the process of making hard copies. Data will only be saved if;

(a) figures have been altered or added.
(b) the advance year facility has been executed.

Printing is straight forward, and if you have not switched the printer on, the program will tell you to switch it on. You can return to the main program or end which will result in a cold start of the computer.

## ACTUAL

This displays the spreadsheet you are viewing. There are two spreadsheets, as explained. The word "estimate" replaces "actual" when the estimated spreadsheet is displayed, and vice-versa. This is displayed everytime you use the spreadsheet, so as not to get confused when amending figures. All of the above commands are available in estimated and actual mode.

## GETTING STARTED

When starting off with BUDGET CALC only the listings presented here will be on disk. The data files have to be created before any of the program will function. The small programs BUD DATA CREATE, BUD SUB CREATE, BUD MON CREATE are those that I actually used when wanting to make the individual files.

## BUD DATA CREATE

This file will create sequential data for BUDGET DATA on disk. This is the longest file that has to be made.

**The variables have the following meaning:**

**SR year advance status**
**CM present year - 3**
**Y$(a) dim of estimated figures**
**H$(a) dim of estimated figures**
**SR,SS,SU are strings which contain last three years monthly totals.**
**BA,BB,BC are strings which contain the last three years subject totals**
**GT(1), GT(2), GT(3) contain the last three years actual totals**

## BUD SUB CREATE

This file will create the data for BUDGET
CH$(a) is a DIM containing an unpadded string
NB and NC contain the number of spaces needed to pad the start and end of the string.

## BUD MON CREATE

This will place the data for the file BUDGET MONTHS on disk.

The variables M$(1) to M$(12) simply contains the month names.

NOTE The above are only needed when creating the files for the first time.

## BUDGET CALC PROGRAM DESCRIPTION

| LINE | DESCRIPTION |
|---|---|
| 100-420 | Printing of screen and input of present month. |
| 430-480 | Dimensioning variables and strings. |
| 490-500 | Loading actual and estimated figures. |
| 510-580 | Loading past figures and totals. |
| 590-640 | Loading subjects. |
| 650-690 | Conversion of zero figures into —. |
| 700-710 | Conversion of figures to spreadsheet variables. |
| 720-730 | Loading month sequence. |
| 740-990 | Setting variables and strings. |
| 1000-1380 | Printing of the spreadsheet. |
| 1390-1630 | Boundary limitations. |
| 1640-2090 | Amend routine. |
| 2010-2550 | Movement of command cursor. |
| 2560-2910 | Subroutine for calculating subtotals for actual figures. |
| 2920-3190 | Subroutine for calculating subtotals for estimated figures. |
| 3200-3460 | Advance year routine. |
| 3470-3490 | Loading of other programs. |
| 3500-3640 | Conversion of variables to the spreadsheet. |
| 3650-3930 | Saving of important variables to be loaded again. |

## BUD REVIEW

| LINE | DESCRIPTION |
|---|---|
| 100-370 | Loading of important variables. |
| 380-550 | Converting actual figures to past figures. |
| 560-780 | Menu. |
| 790-980 | Review subjects (1). |
| 990-1180 | Review subjects (2). |
| 1190-1350 | Review months (1). |
| 1360-1510 | Review months (2). |
| 1520-2150 | Graph 1 (0-4000). |
| 2160-2800 | Graph 2 (0-9999). |
| 2810-3260 | Profit and loss screen. |

## BUD END PROGRAM DESCRIPTION

| LINE | DESCRIPTION |
|---|---|
| 100-350 | Loading of important variables. |
| 360-380 | String defining. |
| 390-750 | Save. |
| 760-960 | Menu. |
| 970-1040 | Test to see if printer is on. |
| 1050-1170 | Inform user that printer is not on and systems reset. |
| 1180-1200 | Branching of different hard copy. |
| 1440-1650 | Estimated figures hard copy. |
| 1660-1920 | Actual and estimated figures hard copy. |

# ADVENTURE WRITING

**JASON FINCH provides another dose for all you Adventure freaks hoping to write your own games**

**Hello once again and welcome to the section of CDU that tells you everything you need to know on how to write an adventure game in BASIC for your C64 computer. Many of you have written asking for copies of the picture files that you have missed from past disks and these requests are being processed at the moment so please be patient. I have made up a disk of all of the pictures, both past and future, that anyone can obtain free of charge by sending a blank disk to: AW PicDisk, 11 Cook Close, Rugby, Warwickshire, CV21 1NG. At this stage I must thank Doug Sneddon, my mate down there in Warminster, for drawing such excellent pictures for use in the demo adventure that I shall provide you with towards the end of this series. Now, onto the business of this month - messages and computer responses.**

## DEFINING MESSAGES

If you want to make your life a great deal easier then every response that the computer makes should be defined as a separate message. These messages can then be recalled as many times as needed in whatever situations arise by using a simple PRINT statement. The maximum length of a message if it is defined by a variable (like M$) would be 255 characters. This is quite adequate for most needs and if you require something a bit longer then just print two messages one after the other. "You poke the guard dog with the long stick, causing him to turn his head slowly and look in your direction. Saliva dribbles from his mouth and he starts to growl ferociously. You decided to play chicken and run off into the stable standing to the east." That last section in quotes resembles a message that you may want to be displayed. It is reasonably detailed and contains a lot of information - still it falls within our restriction of 255 characters per message. So how exactly do you define these messages? Well you need to READ in all the information from DATA statements. Strictly speaking you don't HAVE TO do it that way but it is by far the easiest - even if it will take a little while to set up. You should know that BASIC lines can only be eighty characters long, and so a method must be devised to link the short DATA statements into longer message "strings". This is done with a routine similar to that shown below:

```
10      FOR X=1 TO 10
20      READ A$: IF RIGHT$(A$,1)="*" THEN 40
30      M$(X)=M$(X)+A$: GOTO 20
40      NEXT
50      DATA "The farmer looks at you with"
60      DATA " eyes that seem to be so penetrating.*"
70      DATA "You walk towards him*"
80      DATA ... rest of info ...
```

In the above routine, the ends of the messages are defined by asterisks and the message strings are built up until the end is detected. The same method can be used for defining the location descriptions.

## THINGS THAT FOLLOW

In a recent article I mentioned that you may want to have other "creatures" following you and that messages like "The goblin follows quietly." may need to be displayed. Now I shall tell you just one method for displaying such messages when they are needed. Assume that message 53 is "The young boy follows you cautiously" and that the computer sets a "flag" (defined by the variable BF - boy following?) when something has been done such that the boy is made to follow you. The BASIC line to print the message is overwhelmingly simple:

```
1000 IF BF=1 THEN PRINT M$(53)
```

That is quite easy to understand but can be developed further if you wish. For each condition you may want a different message. It is best if you plan all these beforehand and type them all in in a single session. You may end up with over 200 messages and you will need to keep a record of what each one says! Otherwise you may find that "The postman shrugs his shoulders" when you ask the chemist for a headache tablet!

You may decide that you want to make your adventure a bit special and don't want to rely on the bog standard PRINT command. In that case, you need to write a subroutine that displays the messages in your chosen fashion and then returns to the main program. Using the above example, it would become something like:

```
1000 IF BF=1 THEN M=53: GOSUB 5000
```

where lines 5000 onwards would display a message defined by the number held in the variable M.

## GIVING HELP

If you decide that you want to dole out a bit of help if the player enters HELP or CLUE then these are also messages and need to be defined. Never give too much help or the player will start to rely on it. Try adopting some sort of code that makes the help messages appear like SFBE UIF NBHBAJOF. Because of the flexible method of using string variables, you can always change them to something readable if, for example, the player has found a magic reading potion.

## OVER TO YOU

I have tried to make this series something that is quite "open-ended" so that there is plenty of scope for reader input. If you have any ideas about adventure games that could be attempted, or you have any hints and tips of your own, then please don't hesitate to drop us a line at Adventure Writing, 11 Cook Close, Rugby, CV21 1NG. I hope that you have found this relatively short article quite helpful and I look forward to reporting on any letters that I receive in time for the next edition of Adventure Writing in the November 1991 issue. See you all again then!

# C64 AUTOSAVE

**One of the most enduring problems which programmers face is the loss of hours of work through human or computer error. With this program you'll be able to save time and a lot of frustration -**
**PAUL EVES**

**You are busy typing in your huge adventure game in Basic. You have been typing for over an hour and decide that at the end of the next line, you will save out your work. Suddenly, a quick power surge resets you 64 and you are left with nothing. (Unless of course you happen to have an OLD program sat somewhere in memory - and how many of us bother to do that).**

That's just one of the hazards of Basic programming. Hopefully, this little program will come to your rescue.

## LET THE C64 WORRY

Before you start typing your Basic program in, load up my AUTOSAVE program first. Type NEW then activate with SYS52480. Now you don't have to worry about continually stopping to save out your work. Every five minutes your work will be saved out automatically, under the name of PROGRAM. Once the program has been saved control is returned back to you, as if nothing had happened. Just prior to the SAVE, you will hear a warning tone, this is so that you can stop typing whatever your're typing and wait for the SAVE. So how does the program work?

## THE TECHNO BITS

As you know, every sixtieth of a second the CPU stops whatever it's doing and scans the keyboard and STOP key. Luckily for us, the vector that points to the interrupt routine lies in RAM. This means, of course, that we can change the vector address to our own address. Therefore, every sixtieth of a second the machine will do what we want it to do before executing its own interrupt routine. (And how do we change the vector?)

The first thing we must do is disable the normal interrupts so that we can change the vector address. If we neglected to do this, the machine would cause an interrupt within an interrupt and just lock up. After changing the vector to point our own routine, we then re-enable the interrupts. The machine will now stop every sixtieth of a second, perform our amended interrupt, then continue with the normal interrupt.

## THE NEW ROUTINE

So, what is our new routine? Firstly, we count down within a loop for five minutes. (I could quite easily have used the system clock under interrupt for this, but this way is shorter and neater). If it's less than five minutes then the routine breaks out to the normal keyboard scan interrupt routine at $EA31.

Once the countdown has reached the five minute mark the routine jumps to the warning tone routine. Here our friend SID performs three short beeps before jumping to the save routine. The screen is then cleared and the KERNAL routines SETLFS, SETNAM and SAVE are called one by one. To make the program flow easier to follow, I have incorporated a name save for both TAPE and DISK. (Obviously, tapes don't need a name in reality). After the save, the interrupts are set back to our routine once again and away the program goes.

Naturally the program could be expanded to incorporate a number of things. For example, you could have a LIST or a LOAD command put on to the F keys as well as the forced save. Obviously the more things that your interrupt has to perform, the slower the program becomes. (And it's not really good practice to rely too heavily on interrupts).

I must say that I've found this program very helpful indeed. I just hope you'll get as much use out of it as I have.

# FILE MENU

## A collection of Basic utilities for the Basic programmer - JOHN CAMPBELL

Last month we gave you the main MENU program and the first 3 of the 10 utilities that make up this suite of tools for the Basic programmer. This month we present you with FILE SEARCHER, FILE MERGER and FILE REPLACER.

### WHAT THEY ARE ABOUT

The File Searcher program allows the Basic programmer to specify a file or series of files, and a character string to search for. Then all lines containing that string in the file(s) are displayed on the screen or printed. This capability is useful as a debugging tool to discover and correct variables assigned the same name but performing different functions, or to list all places where a given statement is used or subroutine is called. It can also be used as a precursor to File Replacer (which searches for a string then replaces all occurrences with another string), by displaying all lines where a given string will be replaced.
When you select the File Searcher utility from the Menu, it is loaded and run. The utility first asks you to supply the name of the existing Basic file where the string is to be searched for.

#### NAME OF INPUT FILE?

You enter the name of the disk file where your program is stored and press the RETURN key. If the file name contains wildcard characters (* or ?), File Searcher searches each matching file name on the disk. The wildcard characters are interpreted by File Searcher the same way as they are interpreted in a disk command. The "?" represents any single character, and the "*" represents a string of characters of any size or composition.
Next, the utility asks you to specify whether the lines are to be displayed on the screen or printed on your printer.

#### OUTPUT TO SCREEN OR PRINTER?

You enter S and RETURN to have the lines displayed on the screen, or P and RETURN to have the lines sent to the printer (make sure the printer is ready).

The lines displayed or printed will contain the string you specify in response to the following prompt:

#### STRING TO SEARCH FOR?

You enter a string of characters exactly as you want them to be searched for. The string cannot contain a carriage return, since the RETURN key signifies the end of the string to File Searcher. In addition, if you have any Basic keywords or operators in your string, you need to enter a "|" symbol as the first character in your string. This character serves only to indicate to File Searcher that the string should be interpreted as a Basic expression; the | will not be part of the search string.
Let's take an example. If you want to find all lines containing PRINT keywords, you need to specify |PRINT as your search string. However, this will not display any REM statements which contain PRINT in the comment, nor will it list any lines where PRINT is part of a string constant enclosed in quotes. For that, you would specify PRINT as the string to search for. Without the | character in front, File Searcher interprets it as a string of characters containing no Basic keywords. Since many Basic keywords are common English words, you need to be aware of the distinction when specifying your search string. Consult your Commodore 64 Programmer's Reference Guide for a list of Basic keywords.

Expressions containing Basic operators must be treated the same as Basic keywords. You must enter the | character first any time you want to search for a string containing *, /, +, -, ~, <, >, and =. If you ever want to see the string interpreted both ways, you need to run File Searcher twice, once with the | and once without it.
Once you have entered all the information, File Searcher begins its work. The utility reads the original program file, searching for the string you specified. As it reads each line, File Searcher updates the display screen to let you know how many lines it has found. It searches each line read for the string specified, and, if a match is found, it displays the line on the screen or sends it to the printer, depending on what you indicated.

You may switch disks if you want to perform a disk search on a different disk, but you must do so before typing in the file name. The program will prompt you to switch the File Menu disk back into the drive before completing the search. To avoid having to switch disks for every new search, you can copy FILE.VER from the File Menu disk to your working disk. This file serves as a reference point to which File Menu returns after each command. You can use any Copy program to copy FILE.VER to your own disks.
There are three possible error messages you may get from File Searcher.

#### 1. ERROR—FILE NOT FOUND

File Searcher could not find the original file from which you want to extract lines. You need to check the spelling of the file name, and make sure that file is on the disk.

Then run the program again with the correct file name.

2. ERROR—NO MATCHES FOUND

File Searcher found no instances of the specified string in the file. Check the spelling of the string you entered. Otherwise, if the search string contains a Basic keyword, you need to specify a | as the first character in the string to have the string interpreted as a Basic expression.

3. ERROR—FILE MENU NOT FOUND

This error occurs when you elect to load the File Menu after completing execution of the utility, but it is not found on the disk. You are prompted again to enter your choice, which gives you the opportunity to insert the proper disk into the drive before responding.

## FILE MERGER

The File Merger program allows the Basic programmer to merge two program files together and create a new file containing the combined program. In conjunction with the FILE EXTRACTOR, FILE DELETER, and FILE RENUMBER utilities, File Merger allows subroutines to be extracted from other programs, renumbered as necessary, and merged into a single program file. Lines are merged together in numerical order, except when both input files contain the same line number. In that case, the output file will contain the line from the second file specified.

When you select the File Merger utility from the Menu, it is loaded and run. The utility first asks you to supply the names of the two existing Basic files which you wish to merge together:

NAME OF INPUT FILE?

You enter the name of the disk file where the first program is stored and press the RETURN key.
Next, the utility asks for the second file to be merged:

NAME OF SECOND INPUT FILE?

You enter the name of the second program file and press the RETURN key.
Now the utility asks you to supply the name you want to use for the disk file to be created to store the merged program:

NAME OF OUTPUT FILE?

You enter the name of the new file and press the RETURN key.

Once you have entered all the information, File Merger begins its work. The utility reads a line from each input file, and compares their line numbers. It writes the lower-numbered line to the output file, then reads the next line from the file where the line just output came from. If both lines being compared have the same line number, the line from the second file is copied to the output file, and the other line is dropped. Then the utility reads new lines from both files. As each output line is written to the new file, the display screen is updated to let you know how many lines have been output.
When the end-of-file is encountered in one of the input files, the remaining lines are copied from the other input file into the output file. When the end-of-file is encountered in the other input file, the output file is closed and processing terminates. The new file contains the merged program which you may load, edit, and run like any other program.

Note to users of the Fast Load cartridge made by EPYX: there is a bug in the cartridge which leaves the computer in a funny state after running File Merger on some files. In that state attempting to load a file starts up but never completes. I have attempted to contact EPYX about this problem, but received no reply. Simply cycling power off and on again clears up the condition. Alternatively, removing the cartridge prior to running File Merger avoids the condition entirely.
There are three possible error messages you may get from File Merger.

1. ERROR—FILE NOT FOUND
File Merger could not find the file which you want to merge with another. You need to check the spelling of the file name, and make sure that file is on the disk. Then run the program again with the correct file name.

2. ERROR—FILE EXISTS
File Merger found a file already existing with the name you want to use for the new file. File Merger cannot replace an existing file. You need to check the spelling of the file name and either delete the existing file or use a different file name for the output file. Then run the program again with the correct file name.

3. ERROR—FILE MENU NOT FOUND
This error occurs when you elect to load the File Menu after completing execution of the utility, but it is not found on the disk. You are prompted again to enter your choice, which gives you the opportunity to insert the proper disk into the drive before responding.

## FILE REPLACER

The File Replacer utility allows the Basic programmer to specify a character string to search for in a program file, then have all instances found replaced with a second string. A new file is created which is identical to the original file in all respects except where the new string replaces all occurrences of the old string. This capability is useful when combining different subroutines which use similar variable names. File Replacer may be run to change all instances of one variable to a different name. This utility may also be used to change all PRINT statements to PRINT# statements, for example. It can also be used to remove strings from a file, by entering that string as the search string, but specifying no replacement string.
When you select the File Replacer utility from the Menu, it is loaded and run. The utility first asks you to supply the name of the existing Basic file where the string is to be searched for:

## NAME OF INPUT FILE?

You enter the name of the disk file where your program is stored and press the RETURN key.
Next, the utility asks you to supply the name you want for the disk file used to store the new program:

### NAME OF OUTPUT FILE?

You enter the name of the new file and press RETURN.
Then File Replacer prompts you to specify the character string to search for in the program file:

## STRING TO SEARCH FOR?

You enter a string of characters exactly as you want them to be searched for. The string cannot contain a carriage return, since the RETURN key signifies the end of the string to File Replacer. In addition, if you have any Basic keywords or operators in your string, you need to enter a "|" symbol as the first character in your string. This character serves only to indicate to File Replacer that the string should be interpreted as a Basic expression; the | will not be part of the search string.
Let's take an example. If you want to find all lines continuing PRINT keywords, you need to specify |PRINT as your search string. However, this will not find any REM statements which contain PRINT in the comment, nor will it find any lines where PRINT is part of a string constant enclosed in quotes. For that, you would specify PRINT as the string to search for. Without the | character in front, File Replacer interprets it as a string of characters containing no Basic keywords. Since many Basic keywords are common English words, you need to be aware of the distinction when specifying your search string. Consult your Commodore 64 Programmer's Reference Guide for a list of Basic keywords.
Expressions containing Basic operators must be treated the same as Basic keywords. You must enter the | character first any time you want to search for a string containing *, /, +, -, ~, <, >, or =. If you ever want the string interpreted both ways, you need to run File Replacer twice, once with the | and once without it.
After you have entered the search string, File Replacer prompts you for your replacement string:

## STRING TO REPLACE WITH?

You enter the character string to use to replace all instances of the search string found. Again, if the replacement string is part of a Basic expression, it must be preceded with the | character. Be careful when replacing shorter strings by longer strings not to exceed the 80-character maximum length of a Basic program line. If that happens you will see a warning message and you will need to correct the file by hand, splitting the long line into two smaller lines.
Once you have entered all the information, File Replacer begins its work. The utility reads the original program file, searching for the string you indicated. If the search string is not found in the line read, File Replacer copies the line unchanged into the output file. If the search string is found in the line, that string is removed from the line and replaced by the replacement string. All occurrences of the search string in the line are replaced before writing the line to the output file. As it outputs each line, File Replacer updates the display screen to let you know how many lines it has processed.
There are four possible error messages you may get from File Replacer:

### 1. ERROR—FILE NOT FOUND

File Replacer could not find the original file from which you want to extract lines. You need to check the spelling of the file name, and make sure that file is on the disk. Then run the program again with the correct file name.

### 2. ERROR—NO MATCHES FOUND

File Replacer found no instances of the specified string in the file. Check the spelling of the string you entered. Otherwise, if the search string contains a Basic keyword, you need to specify a | as the first character in the string to have the string interpreted as a Basic expression.

### 3. WARNING—LINE n EXCEEDED MAX LENGTH

File Replacer detected a possible line length overflow when replacing a shorter string by a longer string. Load the output file and list the line number indicated to determine whether or not you need to edit the line or split it into two lines.

### 4. ERROR—FILE MENU NOT FOUND

This error occurs when you elect to load the File Menu after completing execution of the utility, but it is not found on the disk. You are prompted again to enter your choice, which gives you the opportunity to insert the proper disk into the drive before responding.

# POP-UP MENUS

**LIONEL JACK provides a handy menu routine for C64 owners.**

You've all seen at some time how POP-UP MENUS can really give programs a look of professionalism. Well now you can achieve the same affect in your own programs. This utility allows the user to call a number of pop-up menus of his/her own design to enhance their own screen display which is returned to its original state when the menus are erased. Although the work horse of the program has been written in machine code (for obvious speed) it has been designed to be accessible from Basic.

## DEMONSTRATION

This routine has been written on a very simple level just to give you an idea of how everything works. The number, size and colour of your own menus is completely under your own control. Refer below for a complete breakdown of the demonstration program.
You will find a short delay when you run the demo. This is just the computer reading the text into memory, this means that the menus will appear almost immediately when called later in the program.

| LINE | DESCRIPTION |
|---|---|
| 5 | LOAD M/C (change device to 1 for tape). |
| 10 | N= Number of menus. |
| 20 | POKE 679,0 to protect original screen. |
| 30-50 | Set up parameters for each menu. |
| 60 | POKE 679,1 to restore original screen. |
| 100-140 | Pass parameters and call M/C to print menus. |
| 200 | Convert menu address to POKE numbers. |
| 500-570 | Read text for menus and store in memory |

## DESIGNING YOUR OWN MENUS

When designing your own menus there are just five parameters that you must pass to the M/C to get the size, colour etc. that you require. If you examine the demonstration program these parameters are:
L - The number of lines that you want in your menu. Always add two to the total that you require to allow for a border across the top and bottom.

W - This is the width of the menu. Again add two to your total to allow for a border down the sides.

C - This is the colour of the shadow of the menu. The program will automatically make the foreground of the menu the next higher colour in the Commodore colour table.If you therefore choose 0 (black) as the shadow then the foreground will be 1 (white).

B - This indicates which menu you wish to display. I'll discuss text later but for now all that you need to know is that the menu numbers correspond to the order in which the text appears in the DATE statements (lines 500-570 in the demo). In other words if B=3 then you will get the third block of text in the menu.

SA - This is the start position of the top left hand corner of the menu. It should be equal to a screen memory location. For your information location 1024 is at the top left hand corner of the screen,you should be able to work the rest out from there.

## ENTERING TEXT

The only other thing you need to know is how to set up your text. You may have as much text as you like in each of your menus, as long as it will all fit on the screen, but care needs to be taken that it appears exactly how you want it to.
When writing your text as DATA statements (see demo) you might have to place extra spaces between words or even have words connected (see line 570 in demo). This is only to make sure that the text is correctly spaced in the menus. Plan your menus carefully before you enter the DATA statements and you should have no problems.
You may have noticed that the text data in the demo has an asterisk (*) at the end of each line. This is merely an indication to the computer that this is the end of the text for a particular menu. As my demo is pretty short my text for each menu has not gone over one line, but as I suggested earlier your text may be as long as you like and may therefore require more than one line to get it all in. Make sure that you place an asterisk at the end of each menu.
Of course if you wish to use the asterisk within your text you will have to change the end of the menu signal to another character. A good character to use is '@'. If you use this change the 52 (character code for '*') in line 530 of the demo to 64 (character code for '@').

## ON YOUR OWN

Remember to always protect your current screen (line 20) before printing a menu and recover it (line 60) when you want to clear the menu.
Do not be intimidated by this explanation, it really is quite simple to produce your own professional looking pop-up menus and it's certainly worth the effort. you can see how short the demo is so try it first and refer to it when you write your own routines.

# HACKMAN & Sobbin 2

**That East Anglian computer widow, JENNI SIMPSON, continues with her sob story**

**Nah! Seriously folks, I didn't really sit and watch the hands of that blasted pine clock on Mal's oh†so boring wall, crawl at the pace of a sleepy sloth, and believe me, I didn't really feel that compared to this afternoon's entertainment, that compulsory viewing of Sir Harry Secomes 'Highway', would have been abundantly more preferential to remaining one more moment in that bleak and alien world of computerese and uncuddley space age gadgets. But fate must have sensed that I could have endured no more. For just as I was about to hack myself to death with Mally's grapefruit knife, (no please don't take the pith), I heard that heavenly exclamation, and just as an alcoholic, free from the clutches†of Betty Ford, or an obese chocoholic, escaped from Weight Watchers, I drank in that oh, so, sweet†nectar†of those beautiful and inspiring†words, until I grew†quite heady†with their meaning... it was time for tea, and subsequently, we were going home. Oh praise the lord and hallelujah!**

The 'techno-two', now upstanding†and engaged once more in a tete a' tete, this time concerning the benefits of having twin disc drives and a hard disk, they ambled like two broody mother hens into Mally's second†'puter' room, to immerse themselves in yet another dimension within the obsessional art of computing.

Pssst... What was that noise? At first, I thought that one of Mally,s balloons had sprung a leak, but no, I heard the impolite request for attention again. (To be read in a Mexican bandido accent, or an extra from one of Flint Breezeblocks movies). "Hey Senorrrita. Come over here. You lika to perrruse my strrrong and attrrractive†boady? What you say to prrressing yourrr tenderrr fingerrrtips gently upon my sensitive keyboards and softly carrressing my harrrdware with yourrr beautiful corrrnflower blue eyes?

Inexplicably, I suddenly felt the whole of my being become acutely sensitised. For the only way in which I am able to describe this curious and thought-provoking phenomena, is I feel, to conjure with the humble adjective. The very meaning of which,†can, I believe, only be interpreted by your good-selves in accordance with your own life experience. However, I shall endeavour to give it a whirl. Firstly, my body began to rapidly vibrate. Then,†strange, hot tingling sensations began to traverse my pulsating veins,†and accumulate within my fingertips, which gradually became completely numb. Tiny beads of perspiration seeped through my clothing, and my heart began to swiftly palpitate. As if suddenly turned on by a mains switch, my aural and visual senses became acutely heightened, and I felt as if I had been 'super-charged'. Pure, unadulterated energy literally rippled, crackled and fired through†my 'highly tuned' body, and I felt all powerful, super human. For even the bionic woman,†(and I don't mean Mrs. Thatch or Dame Edna), could not, I feel sure, have held a candle†to the way that I was feeling at that very moment in time.

And then, just as unexpectedly as this weird phenomena had begun, it abruptly ceased. The illusion of time slurred, staggered, and then carelessly lurched to a faltering and unsteady halt. Chameleon-like, my awareness shifted once more, and I became unaccountably and unashamedly aroused. My very essence tingled with intense tremors of excitement and expectation as I became irresistibly and uncontrollably drawn toward the Amigo (whoops, sorry, the Amiga). Then, in a dreamy spaced-out, trance like state, almost as if under deep hypnosis, I found myself upon my feet and floating in slow motion, as if under water, toward that 'infernal machine'.

Now, hovering directly in front of it, I rubbed my eyes in sheer amazement as I gazed upon the vision which was now before me. For there, glowing in a spectacular, shimmering and pulsating cascade of iridescent, fairy tale hues, was the machine. Suddenly, the incredulous display of glittering rainbow lights had ceased, and I found myself seated before the computer. "Please Senorrrita," again to be read in a Mexican bandido accent, "please to touch me.....I beg of you". Was this an aural hallucination induced by my fevered imagination? Had the Amigo actually spoken to me? Or had I just got wax in my ears? Which ever it was, I felt an urgent compulsion to place my fingertips upon the awaiting keyboard. But just as I was about to place flesh upon plastic, another strange occurrence caused me to rapidly retract my now doddery digits. For out of the mouthlike orifice of the Amiga's disk drive, oozing like some ectoplasmic slime from 'Ghostiebusters', were countless, swirling, phosphorescent tendrils of verdant, vaporous mist.

Reaching out toward my rigid, petrified body, they curled like indian smoke signals, to gently entwine and embrace me within their sensual dance of 'ultimate knowledge'! All of my anxieties now melted as snow in warm sunlight, and a wondrous feeling of wellbeing coursed through my joy-filled psyche like a soothing river of molten gold. Peace and tranquillity radiated throughout my being, and a pristine mirror-like clarity replaced the jumbled maze of confusion that once had been my chaotic thought process. Clutter-free, and totally devoid of all random thought, my mind, like a highly absorbent sponge, began to assimilate the potent energies which were summoning me. Then, just as a moth is driven toward the light, my hands reached for the Amiga keyboard, and my fingers, as sensitively as a blind person's interpretating braille, found and caressed those small, magical indentations with a such a supreme pleasure, which was, I believe, way beyond any form of description. Now, likened to young, rampant lovers, separated for many weeks, Doggy and the Amiga came together urgently, to passionately consume their charismatic relationship in a harmonious and blissful merging of woman and machine.
Oh sweet, sweet, sweet joy. The contentedness of Buddha was now mine! Indeed, this was no 'Dogma' (ho! ho! ho!). For knowledge and logic, like the slow realisation of an intangible concept, began gradually to rain upon the virgin grasslands of my mind. And growing insight into the world of computing became stimulated and richly nourished, as information and enlightenment continued to pour steadily onto the awaiting landscape of my consciousness. Just as parched and dust ridden soil readily drink droplets of moisture, so too did my being gulp greedily from this infinite cornucopia of knowledge.

Great waves of multi-parallel, binary coded data, flooded and burst relentlessly through my neural processor, exciting and teasing the vast arrays of my tri-state neuron receptors. Breathtaking fractals whose brilliant colours, all too numerous to mention, appeared suddenly before my eyes, before leading me, like a shot from a twelve bore, down and down, deeper into their infinite and awesome depths. My body felt hot and glowing, as if it were on fire. Vibrating with an overload of purest energy, it convulsed in response, as a multitude of tiny tremors, like the transient scintilla from a child's sparkler on Fireworks Night, penetrated every fibre of my being, to leave me feeling 'high', 'alive', and 'wired'. I suddenly knew that I was being prepared for something awesome, something beyond all comprehension.

I was being purified, primed, being made ready. But for what exactly, and most importantly, why? With these curious thoughts and feelings buzzing around my mind, filling me full to bursting with their countless, mind boggling possibilities, and my tense, ready to go body urging me to fulfill it's physical need for action, I was rudely interrupted and immediately distracted, when Boney and M, decided to wander, like an old married couple, back into the room. Muttering between themselves like two senile delinquents, they insensitively edged me away from my Amigo, as if I were an old table in urgent need of being cleared.
"Been playin' 'Chuckie Egg'," enquired Bono, with a patronising grin, as he sat down heavily in MY seat.
"Nah," I replied, with a shake of my head. "I have just been given the powers of computer genius, and was about to have the meaning of life divulged to me, that's all!"
"Mmm, good," replied the Bono, not listening to a word that I had just been saying.
"So Mally," he continued, as if I had simply vanished into thin air, "if we load up the DOS2DOS, we shouldn't have any probs downloading the text files from your IBM to the Amiga!"

# BASICS OF BASIC

## JOHN SIMPSON continues his series for beginners to Basic

**Last month we looked at the PLOT routine for doing some pretty neat tricks with the cursor. As promised, this month we will be looking at some routines for SORTING and SEARCHING. There is quite a lot of text so I hope you don't get bored.**

### INTRODUCTION

We often find the need within a program to sort tables into numerical or alphabetical order. Score tables in games, names in data bases, for example. Once we have sorted them, we usually find it necessary to access them, and if you have a data base with a thousand names within it and you are looking for one name in particular, then a fast SEARCH program become a necessity. We don't want our user to nod off to sleep, do we? However, we shall tackle sorting first.

### SORTING

Imagine if Aunty BT sent our telephone books to us without having them listed neatly into alphabetical order! What an incredibly difficult task it would be to find someone's telephone number. Let's make a modest start by writing a small program which will request the input of two names and then sort them into alphabetical order.

**This, then might be our algorithm:**

1. Program Start.
2. Input Name #1
3. Input Name #2
4. If Name #1 is lower alphabetically than Name #2 go to 6 otherwise go to 5
5. Output name #2 - Name #1 - end
6. Output Name #1 - Name #2 - end

Step 4 begs a question, How do we determine a name to be alphabetically lower than another name?

As you know, our computer only understands numbers, and letters are really the representation of coded numbers - remember ASCII! So it is quite easy to compare groups of letters held in strings. Thus if

```
A    $ = "DOG"
B    $ = "CAT"
C    $ = "MAN"
D    $ = "DOGS"
E    $ = "MAN"
```

then C$ = E$ - the same word
but A$ > B$ - Dog is further along the alphabet than Cat
and D$ > A$ - the extra 's' places it after 'dog' in alphabetical order

**So, now to write our program.**

```
100 REM ALPHA-SORT PROGRAM
110 REM INPUT SECTION
120 INPUT "[CLR]FIRST NAME?";A$
130 INPUT "SECOND NAME?";B$
140 REM TEST AND OUTPUT SECTION
150 IF A$<B$ THEN 190
160 PRINT "FIRST  "B$
170 PRINT "SECOND "A$
180 END
190 PRINT "FIRST  "A$
200 PRINT "SECOND "B$
210 END
```

Sorting two names, such as in the foregoing program, is relatively simple, but what if we want to sort three names or more? For this we have to stop and think - and the first step of our thought process will usually bring us back to Arrays. We need to input our information and then store it in an array. So the very first thing is to DIMension an array, then write an input algorithm which will gradually fill the array. This will be followed by a sort routine to adjust the array into descending order, alphabetical or numerical, after each input, or by the use of a requester.

A neat hypothetical situation could very well be a game environment program where you want to keep a High Score Table. Let's say we want to store and display the six highest scores so far. Something which may look similar to this;

**HIGH SCORES**

| | |
|---|---|
| 1 Benny | 50000 |
| 2 Jumping J Flash | 49320 |
| 3 Harold | 36489 |

Etc. Etc.

Obviously the names are not in an alphabetically ordered display but the numerical scoreline is. In this situation we will require two arrays; a string array for the names and a numerical array for the actual scores. We will need to sort the numerical array, and force the string array to follow the numerical. At the start of the game the array will be either empty or full of your own high scores until gamesters have achieved their own. You can construct a save and load

option into the game to keep a high scores file permanent, or not.

The next piece of code we will require is a variable which will contain the current score of the player of the moment. Once the game ends we will need to check this Current Score against the scores in the score array to determine if this one merits a place on the High Score Table. If not then simply end the game but if it does, then we need to input the player's name (if we haven't already done so), place the score and name into the two arrays, sort them into order, and, finally, print the result to the screen with or without an option to save the scores file

## THE HIGH SCORES PSEUDO-GAME

Now we shall commence to put all of this into practise with a simulated game. In this 'game' all that will happen is a random number will be generated to represent the current player's score. We shall construct the game around a 'main' processing loop. It is from within this loop that all the necessary subroutines will be called to deal with each situation as it occurs. The following is the program finished except for the final task, which would be to renumber the program, and if desired remove all the REMs. I have given an outline of each section followed by the lines of code then, at the end of the program, I take a look at each section dwelling upon the more important aspects in some detail.

As well as developing a sort routine within a pseudo-game we will also be looking at the aspect of program construction adopting a method know as 'structured programming'. This is somewhat of a misnomer for Basic programs because, for reasons I will deal with at a later stage, Basic is not really a suitable language for such a method - however, we can approach a reasonable degree of 'structure'.

The following, then, are the basic sections which go to make up a typical program.

**1. Start of the program.**
**2. Call to SETUP from where we define and initialise program variables, dimension and fill the arrays, print our starting screen, etc.**
**3. MAIN loop where we call all the necessary routines, etc.**
**4. The various subroutines used to execute the program.**
**5. SETUP routine.**
**6. DATA Statements.**

### 1. PROGRAM START

The very first item in the code should be your copyright declaration, date, and your name (address and phone number if you so desire).

```
10 REM ************************
11 REM * MY GAME            *
12 REM * COPYRIGHT (C) 10/9/91*
13 REM * A CDU READER        *
14 REM ************************
15 :
```

Note the use of the line (15) with just a colon. When I want to open up some space in the program to highlight each section, or whatever, I tend towards using colons instead of REMs - it looks a little more neat both for yourself when developing the program and for any other person who needs to look at, and make sense of, your code.

### 2. CALL SETUP

I usually reserve line numbers 50000 to 51000 for my setup routine - This is nicely out of the way and eventually, when the program is complete, then I will renumber the entire program and take out the REMS and COLONS only lines for my final version

```
18 REM *** THE SETUP ROUTINE ***
19 :
20 GOSUB 50000
30 :
```

### 3. MAIN

In a complicated game this loop would tend to be fairly large, but in this program it will not be because a general game is absent. However, I like to set aside line numbers 100 - 2000 for this section. The final line of this section is an unconditional jump (GOTO) which redirects program control back to the start of MAIN. The program will continue looping around this section until told otherwise, such as when the game concludes.

```
98 REM *** MAIN PROCESSING LOOP
99 :
100 PRINT"[CD]WANNA QUIT Y/N?":GOSUB6100
105
IFF3=0THENPOKEB0,14:POKEB0+1,6:PRINT"[CLR][LBL
UE]":END
110 GOSUB 2000
120 IFF1THENPRINT"[CD][GREY3]SORTING
SCORES...":GOSUB3000
130 PRINT"[CD][YELLOW]VIEW LEADER BOARD - Y/N
?"
140 GOSUB6100
150 IFF3THEN300
160 GOSUB3200
170 GOSUB6200
300 GOTO100
999 :
```

### 4. SUBROUTINES

Normally, with a game, I would be using various subroutines, such as Joystick, Keyboard input, Output, Sprite Control, Animation, Sounds, Collision detect, Panel or screen updates, etc. I would locate each routine between the end of MAIN and beginning of SETUP. However, because this 'game' only consists of a randomly generated High Score value then only a few routines will exist, namely 'RANDOM' and 'HIGHSCORE', plus a DELAY routine, a REQUESTER routine, and an input (GET) routine.

```
1998 REM *** THE RANDOM ROUTINE ***
1999 :
2000
PRINT"[CLR][CD3][CR3][WHITE][RVSON]GETTING A
RANDOM SCORE...[RVSOFF][CD3]"
2010 FORX=0TO23
2020
CS=INT(RND(0)*1000):PRINT"[GREEN][CU][CR7]>>" CS
"[CL]"
2030 IFCS=>SC(5)THEN2080
2040 PRINT"[CD]SORRY" CS "IS TOO LOW FOR
LEADER BOARD!"
2050 GOSUB6000
2060 F1=0
2070 RETURN
2080 PRINT"[CLR][CYAN][CD2][CR]CONGRATS! YOUR
SCORE IS" CS "AND YOU"
2090 PRINT"[CR]HAVE A POSITION ON THE LEADER
BOARD"
2100 INPUT"[CR6]PLEASE TYPE YOUR
NAME";NN$:F2=0
2110 IFLEN(NN$)>9THENPRINT"[LRED]NAME TOO
LONG! MAX 12 CHARS ONLY":F2=1
2120 IFF2THEN2100
2130 SC(5)=CS:NM$(5)=NN$:F1=1
2140 RETURN
2150 :
2998 REM *** SORTING THE HIGH SCORE ****
2999 :
3000 FORJ=0TO6:SD(J)=0:ND$(J)="":NEXT
3010 FORT=0TO5
3020 ST(6)=SC(T):NT$(6)= NM$(T)
3030 FORJ=6TO1STEP-1
3040 IFST(J)<ST(J-1)THEN3080
3050 TP=ST(J-1):TP$=NT$(J-1)
3060 ST(J-1)=SD(J):NT$(J-1)=NT$(J)
3070 ST(J)=TP:NT$(J)=TP$
3080 NEXTJ,T
3090 FORJ=0TO5:SC(J)=ST(J):NM$(J)=NT$(J):NEXT
3010 RETURN
3120 :
3198 REM *** PRINT HIGH SCORE TABLES ****
3199 :
3200 PRINT"[CLR][CD3][CR4][LRED][RVSON][CR]THE
HIGH SCORE TABLE[CR] [RVSOFF][CD][WHITE]"
3210 FORJ=0TO5
3220 PRINT"[CR5]"J+1,N$(J),SD(J)
3230 NEXT
3240 RETURN
3250 :
5998 REM *** DELAY ROUTINE ***
5999 :
6000 FORX=0TO2000:NEXT
6010 RETURN
6020 :
6098 REM *** USER INPUT ROUTINE ***
6099 :
6100 GETA$:IFA$<>""THEN6100
6110 GETA$:IFA$=""THEN6110
6120 IFA$="Y"THENF3=0:GOTO6150
6130 IFA$<>"N"THEN6110
6140 F3=1
6150 RETURN
6160 :
6198 REM *** SAVE FILE REQUESTER ***
6199 :
6200 PRINT"[CD]DO YOU WANNA SAVE THE TABLE
TO DISK Y/N?"
6210 GOSUB6100
6220 IFF3THEN6250
6230 PRINT"[LBLUE]SAVING TO DISK-FILE" :PRINT
"PAUSE...":GOSUB6000
6240 REM *** SAVE ROUTINES CALLED FROM HERE
6250 RETURN
```

## 5. SETUP

This is the first routine which the program will call, and only once. Some programmers place this routine at the very beginning of their program because it is a routine which is usually only used once during the operation of the program. I like to place it at the end, or if I am using data statements, just in front of the data, because then I can add new statements to it as my program develops without worrying if I've got enough room before MAIN starts.

```
49998 REM ***  SETUP PROGRAM ***
49999 :
50000
BO=53280:DIMSC(10),ST(10),NM$(10),NT$(10):CS=0:TP
=0:TP$=""
50010 POKEBO,0:POKEBO+1,0
50020 PRINT"[CLR][CD3][CR12][GREEN]HIGH SCORES
DEMO"
50030 GOSUB60000
50040 PRINT"[CLR][CD][LRED]WANNA LOAD HIGH
SCORES FROM DISK Y/N?"
50050 GOSUB6100
50060 IFF3THEN50090
50070 PRINT"[LBLUE]LOADING DISK-FILE" : PRINT
"PAUSE...":GOSUB6000
50080 REM *** LOAD ROUTINES CALLED FROM HERE
50090 FORJ=0TO5:READ
D,D$:SC(J)=D:NM$(J)=D$:NEXT
50100 RETURN
50110 :
```

## 6. DATA STATEMENTS

The final section of my program will contain any data tables that the program may require.

```
59998 REM *** DATA TABLES ***
59999 :
60000 DATA
150,CDU,140,AAA,130,BBB,120,CCC,110,DDD,100,EEE
```

## A CLOSER LOOK AT THINGS

The first part of the program to be executed is the call at line 20 to the SETUP routine located at Lines 50000-50110.

LINE 50000 This line initialises variables and arrays. The program uses and dimensions four arrays, SC() - SCores, NM$() - NaMes, ST() - Scores Temp., and NT$() - Names Temp. CS = Current Score. TP = TemPorary numerical, and

TP$ = TemPorary string. I dimensioned the arrays larger than is currently being used in case I decide to make the Table hold and display 10 High Scores.

LINE 50010 Sets screen and border colours to black.
LINE 50020 Prints the pseudo-game title.
LINE 50030 Calls a delay routine (6000).
LINE 50040 Prints a requester to load saved scores.
LINE 50050 Calls the user input routine (GET A$).
LINE 50060 Evaluates user response by testing the third flag 'F3'of the three flags the program utilises. If flag 'F3'equalled zero then the test would be false and so the ...THEN part of the statement would NOT be executed.
LINE 50070 Prints a message and calls the delay routine of 6000
LINE 50080 From this point we would normally call a routine to deal with loading a file from the disk - we shall be dealing with load/save later.
LINE 50090 This line reads the data lines at 60000 and stores the data into the two arrays SC() and NM$(). Note that the loop is only half the length of the data statements - this is explained by the use of the READ statement which uses first a numerical read, D followed by a comma, then a string read, D$. After which D is stored into SC() and D$ is stored into NM$(). The internal data pointer will then move on to the next pair of datum to be read.
LINE 50100 Terminates the routine SETUP and returns processor execution to line 30 of the program.
**Which brings us to the MAIN processing loop.**
LINE 100 Queries whether the user wants to quit or not, and calls the input routine at 6100
LINE 105 Tests the state of flag 'F3' if it zero, then screen colours are reset to standard and the program ends.
LINE 110 Calls the Random Routine - which replaces a game by simply producing a random number to use as a Score.
LINE 120 If flag 'F1'is true then the score is high enough for the High Score Table so the main sort routine at 3000 is called.
LINE 130 Prints a requester to view the Table.
LINE 140 Calls the input routine.
LINE 150 Makes a conditional jump, if the answer was "N" no, to line 300.
LINE 160 Calls the routine to print the High Score Table.
LINE 170 Calls the Save File requester.
LINE 300 Redirects processor execution back to LINE 100 to continue MAIN.

## THE RANDOM ROUTINE or pseudo-game.

LINE 2000 prints a message.
LINE 2010 Sets up a loop of 24 iterations.
LINE 2020 Generates a random number between 0-999 which is stored into the CS (Current Score) variable, this is then printed on screen. Note the use of the embedded commands to move the cursor back up the screen so that the value in CS is always printed at the same screen location. Also note that after the number is printed an embedded cursor left and several blank spaces have been included to ensure that when moving from a three digit number to a one or two digit number the 'extra' digits do not remain displayed giving an erroneous number - remove the left cursor and spaces, run the program, and you will soon spot what I mean! To finish this line a very short FOR...NEXT delay routine has been incorporated which enables the user to see the numbers as they are generated and printed.
LINE 2025 perform the NEXT of the loop set up in 2010, and when finished calls the delay routine (6000).
LINE 2030 Test CS against the lowest score in the High Score Table (HST). If it equals or is greater than the lowest score then a branch to line 2080 occurs, otherwise
LINE 2040 Prints a message stating that the score is too low for the HST.
LINE 2050 Calls delay routine.
LINE 2060 Sets flag 'F1'to false.
LINE 2070 Returns control back to line 120 in MAIN, with flag 'F1'false.
LINE 2080 - 2090 The score was high enough for entry on HST so print congrats message
LINE 2100 Input the user's name, store it in NN$ variable, and set flag 'F2'to false.
LINE 2110 Tests the length of the inputted name NN$ and if out of range prints a message to this effect, and set flag 'F2'to true.
LINE 2120 test the flag 'F2'and if true branches back to the input Line 2100
LINE 2130 If flag 'F2'is false then both CS and NN$ replace the current data held in the lowest position of the arrays, i.e., SC(5) and NM$(5).
LINE 2140 Returns control back to line 120 in MAIN with flag 'F1'true.

## THE SORT ROUTINE.

LINE 3000 This line fills the numerical array ST() with zeros, and the string array NT$() with null strings.
We need to set up two loops, an outer loop which will, in turn, check every item in our SC() array, and an inner loop which will test the item from SC() array with each item in ST() array. Note that the outer loop iterates from 0 to 5 loops with 'T' holding the value of the outer loop count, and that the inner loop iterates from 6 to 1, with 'J' holding the value of the inner loop count.

LINE 3040 sets up the inner loop. Let us examine this line in more detail, it being the first time I have used this expression: FOR J = 6 to 1 STEP -1. The loop counts down from 6 to 1, the computer understands this by the use of 'STEP-1'. If you put 'STEP-2', then the computer would count down in two's, or 'STEP-3' in three's and so on. By leaving out the minus sign the computer would count up in two's three's and so on, e.g. FOR N = 0 TO 12 STEP 2.

We know that when we first enter SORT each item in ST() is zero, so the first item from SC() must be greater than any item in ST().
You will note that the ST() array has one element more than does SC(), and we will use this sixth (or extra) element within which to place our item from SC().
We compare, on the first iteration of the inner loop, ST(6) against ST(5) and if ST(6) is lower in value than ST(5) then we leave things as they are - however, we know that ST(5) contains a zero (from LINE 3000), and so ST(6) must have a higher value. Therefore we take ST(5) and store it within a

temporary variable (TP), and transfer the value held in ST(6) into ST(5). We then take the value (which was originally held in ST(5)) from TP and transfer this into ST(6). Thus, effectively, toggling the two values around.

<< ST(5) to TP-ST(6) to ST(5)-TP to ST(6) >>

The counter value for the inner loop is now decremented and we do exactly the same thing again. First we compare ST(5) against ST(4). If ST(5) is greater than ST(4) then transpose ST(5) with ST(4):-

<< ST(4) to TP-ST(5) to ST(4)-TP to ST(5) >>

We continue decrementing the inner loop until eventually all elements of ST() have been dealt with. We then increment the outer loop and test the next element of SC() against each element of ST(). This time, however, the highest element in ST() - ST(0) will contain a non zero value, our first value from SC(0), eventually, when the innerloop has iterated to reach this point then whichever is the lower of the two values will be placed within ST(1).
And so on until each value in SC() has been sorted and transferred to ST(). ST() will now hold the updated HST. The final step is at LINE 3100 where the contents of ST() minus the sixth element, the previous lowest score, are transferred back to SC(), thus updating SC().
All the while this is happening, the corresponding name attached to a score is also being sorted and transferred within NT$() in conjunction with ST().

I recommend that if you haven't grasped exactly how the SORT works from my description, then, using pencil and paper, study the code and work out for yourself exactly what it is doing.
You can see why SORT routines are somewhat slow - to sort 6 scores requires 6 x 6 loops of the routine. Imagine how much time a list of 100 values would need to complete! However, let me hasten to add that this is not the fastest of Sort routines - later we shall have a look at two other sorts, a SHELL-METZNER and a TOURNAMENT sort.

## PRINT HIGH SCORE TABLE

LINE 3200 Prints the header.
LINES 3210 -3230 Sets a loop which will print position, name, and score. Note the use of commas to format the print, and the use of the loop counter 'J' plus '1' to print the position.
LINE 3240 Terminates the routine by returning control back to Line 170.

## DELAY ROUTINE

LINES 6000 - 6010 This is a straightforward loop which counts from 0-2000 acting as a short delay.

## INPUT ROUTINE

LINE 6100 Here is something new! After the normal 'GET A$', instead of checking for no input, i.e., IF A$ = "" THEN... We do the reverse by using, IF A$ <> "" THEN... Exactly what this does can be explained by the fact that the 64 incorporates what is termed as the Keyboard Buffer. This 'buffer' holds the last 10 key inputs (characters). So, if the user presses several keys, or even the same key a few times, then when the normal GET statement is used, and the line checks for a response with the usual IF A$ = ""...., this is found untrue because the 1st character is taken from the keyboard buffer, and control slips through to the next line. Now if the first character in the buffer happens to be a character the program is looking for, then it will echo the response, perhaps too soon and when you don't want it to. So, by utilising IF A$ <> ""..... then it will loop around the GET statement until it has cleared, or flushed out the 'buffer'. The next line is always the more usual IF A$ = ""....
LINE 6110 The standard GET statement. We are checking for a "Y", "N" response.
LINE 6120 If the 'Y' key has been struck we set the flag 'F3' to false: and jump to line 6150.
LINE 6130 If the 'N' key has NOT been struck control loops back to line 6110

(NOTE that the true/false flag does not respond with "Y" = true "N" = false key response, it simply makes "Y" false and "N" true to make for quicker program response - it is still 1 = true and 0 = false, however.)

LINE 6140 To reach here the key 'N' has been struck so set flag 'F3' true.
LINE 6150 Returns control to caller.

## REQUESTER ROUTINE

This routine simply requests a response from the user whether or not they want to save the HST to disk.

LINE 6200 Prints the message.
LINE 6210 Calls the input routine.
LINE 6220 If flag 'F3' is true then skip to line 6250.
LINE 6230 the flag must have been false so print the message "Saving..." and "Pause..." and call the delay routine.
LINE 6240 From here we would call our save routines (again, we will come to saving and loading later in the series).
LINE 6240 Returns control to the caller.

## IN CONCLUSION

Examine, and play around with, the ideas and routines used this month until you fully understand them. Next month we will look at some 'faster' sort routines, as well as changing this one to make it run faster. Perhaps you can spot a faster way to perform the sort; considering that the scores are always in order, except, that is, for the most recent! We will examine some search routines which we can use to find important data once it has been sorted. We shall also be taking a look at the joystick and just how easy it is to access it, as well as 'flow-charting' a program.

See you all next month...

SCREEN 17408
CHARS (MAP) $5000
CHARS (STATUS) $5800
$6000 - $8000 available



# EIGHT SPEED SCROLLER

**Learn the art of smooth scrolling with this handy program - RICHARD IKIN**

The routine I have presented here is a hardware scroll routine for the Commodore 64. It is an 8 speed, 8 directional scroll routine. The two speeds (horizontal and vertical) are set at 1 pixel, however these can be altered by changing XSPEED or YSPEED in the assembly language program or, by breaking out of the program by using RUNSTOP/RESTORE and poking the two locations shown below.
The routine works by altering the hardware scroll registers whenever the joystick is used and changing a pointer to the top left of the screen in memory when needed. The screen contents are not moved by a software scroll as usual but the screen is taken from the map and printed onto the video matrix.
The routine is a very smooth hardware scroll routine. I use this routine myself in my own games so I know it works.

## HOW TO USE

Use the joystick in port 2 to scroll the screen. None of the ROMS are switched out, therefore you may notice them on screen, as you will all 64K (in 1K sections).
The scroll moves in the opposite direction to the way the joystick is moved, as it would in a game, i.e. to scroll left, move the joystick right.
The routine presented here is not really a complete program for you to use. What it does do is illustrate how you would produce such a routine for your own programs.

## THE PROGRAM

The eight speed scroller is presented both as a M/C program and as an assembler listing.
As you can see from the assembler listing, the first task is to set the raster interrupt. This is what the "setint" routine does. It also switches the Vic chip to look at memory locations 16384 to 32767 instead of the default (0-16383).
You may notice that the program does not switch out any of the ROMs. The Basic ROM can be switched out by clearing bit 0 of location $0001. However, if you do this you must change the RTS in line 998 to an endless loop, as the RTS will cause a return to Basic out this will cause nasty problems.

OK, whenever a raster interrupt is generated, the 64 will divert its attention to the interrupt routine. The first thing that should be done when the routine is entered is to check the direction of the scroll. This is held in the variable "way". If you look at lines 6160 to 6174 in the assembler listing you will see what values need to be in "way" to cause scrolling. As you can see, a zero will cease all scrolling.
When the direction has been decided and appropriate action taken, the raster split is processed. The comments in the assembler listing explain what action is taken.

Lines 1741 to 2620 are the four routines that perform the actual hardware scrolling of the Commodore screen. These are the routines that provide the smooth part of "Smooth Scrolling".

As you may know, the Vic II chip can alter the horizontal and vertical position of the screen with up to eight vertical and eight horizontal positions. Therefore to achieve smooth scrolling we must:
a) alter the scroll register until it reaches its maximum or minimum value.
b) when this point is reached we must change the address of screen (2) in the map.

After every second interrupt the contents of screen (2) are displayed on screen (1). If you look at the "window" routine you will see that the section of memory/addressed by the 16-bit value "map" is loaded into the video matrix at 17408.
The contents of "map" is the address on the top left corner of screen (2) in the map. Screen (2) is an exact replica of the screen you see when you look at your TV. All it does is point to the data which is displayed on screen.

As the routine will scroll through the whole 64K of the Commodore's memory you will at times no doubt see some pretty weird data printed up on the screen. This is because none of the ROMs are switched out and what is being printed up is the ROMs working storage areas being displayed on the screen. When you set a limit to the size of the scrolling area and switch the Basic ROM out, none of this occurs.

The control routine is a fairly standard routine and can be replaced by your own, as long as you store the correct values in "way".
The speed of scroll is set by the value in "Xspeed" and "Yspeed". The value should not be less than one and not more than seven. They are independent of each other.
In the future I will add a routine to set the size of the scrolling, and will also present any further alterations that I make. I would be interested to see anybody else's ideas.

# BASIC MACHINE LANGUAGE TECHNIQUES

## Part Five of this series gets underway for all M/L novices - John Simpson

**Last month we looked at a simple MACHINE CODE joystick routine. This month we expand on that and provide a simple routine for manipulating a sprite.**

## ESTABLISHING EQUATES

The EQUATE allows the programmer to equate names with addresses or data. This is a pseudo-op and is almost always given the mnemonic 'EQU' or '='. The names may refer to device addresses, numeric data, starting addresses or fixed addresses, etc. Most assemblers allow you to define one label in terms of another, for example:

**FIRST EQU $D000**
**LAST EQU FIRST+$FFF ; I.E. $DFFF**
**or**
**LAST = FINAL ;** This is useful for patching together programs that have different names for the same addresses or whatever. More of this later.

Equates are usually defined at the program header, and if they have not been defined first, and are subsequently used, then on assembly an error code will be generated.

Here is the layout of a typical program:

**(A) Program Organisation - this is where it will sit in memory**
**(B) Assembler directives - such as disable screen listing, or stop assembling if an error is found, etc.**
**(C) Equates**
**(D) Caller for (Setup)**
**(E) (Main) processing loop**
**(F) Various routines called by (Main), and (setup)**
**(G) Tables, data, etc.**

Often, because of memory management requirements, things are not quite so clear cut as this when developing a large program - you will find that 'bits' of your program are spread throughout memory, some portions here, and others there linked by 'jump tables' unconditional jumps and subroutine jumps.

For our small demo program we will only require five equates. These will be:

**(1) The base address for the sprites (53248, or $D000).**
**(2) The base address for the sprite data pointers (2040, or $07F8).**
**(3) The register address of the game port for the joystick (56320, or #DC00).**
**(4) The Kernal CHROUT address for our clear screen function ($FFD2).**
**(5) BUFFER this is and area of free ram which we shall allocate for our sprite image data (832).**

## MANIPULATING A SPRITE

When we come to manipulating the sprite around the screen there are various needs to take into consideration. The first of these is, has the joystick fire button been pressed? If it has then we could call another subroutine which will deal with the fire action before considering anything else. Next we need to consider exactly how fast we desire the sprite to move and so must set a timing counter to take care of this. Obviously we cannot just call some sort of universal delay routine otherwise the whole program would go on 'pause' whilst the delay executed, so it must be an internal (applicable only to that routine) delay mechanism.

Once the delay has been exhausted and the processor is executing the internals of the movement routine then we must take account of directions generated from the joystick, plus the screen's XMSB positions. I'm sure you know what the 'XMSB' is and why we have to take account it, but, just in case you are not too sure, here is a quick run down.

The screen consist of 320 pixels from left to right, and a byte, as you know, can only hold a maximum of 256 bits (0-255) - each pixel is the equivalent of a bit. So, when we require to move

the sprite to the extreme right hand side of the screen, then the sprite X coordinate will require one extra bit to hold the value of the previous 256 pixels, as the Sprite moves beyond 256 pixels - the Sprite's X coordinate flips from 255 to zero. The byte at memory location $D010 is used for this, each bit of the byte is laid aside as that extra (or 8th) bit for each of the eight Sprites. As a Sprite moves to and fro across the screen we must, then, turn on and off its XMSB at $D010.

Here follows a routine, which is very simple and upon which you can dwell and amend to take into account more sprites, possibly sprites moving under 'their own steam' and not from the joystick, etc.

## SIMPLE PROGRAM

```
2000    MOVESPRT INC DELAY ;I have laid aside a variable
        called delay and every  time we enter this routine
        delay is incremented
2020    BNE LEFT1 ;Whilst it is greater than zero it will cause
        a branch to label left1 - which is an exit from the
        routine - hence no action will take place, when it
2030    LDA #192  ;does = o then it will drop through to this
        line where we commence to replace delay with its
        original
2040    STA DELAY ;value ready for the next delay cycle - it
        means that  this routine will activate every 64 cycles
        of main
2045    LDA BUTTON ;Test the fire button
2060    BEQ VERTICAL ;If inactive move to test vertical
        movement
2070    JSR FIREACTN ;Else call fire action - when it has
        finished it returns to vertical so that we can link
        movement and fire together
2075    VERTICAL LDA DELTAY  ;Has a Y Coordinate been
        chosen?
2100    BEQ HORIZON  ;No - so look at the horizontal
        movement
2110    BMI MOVEUP  ;Yes - so is the movement up or
        down? if up then branch to moveup but if down then
        continue with next instruction
2105    ;
2120    MOVEDOWN INC VIC+1   ;Increment the y
        coordinate of the sprite - when it reaches 256 it will
        flip to zero
2140    JMP HORIZON  ;Must jump from here to avoid
        moveup - otherwise we would do an increment
        followed by a decrement!
2145    ;
2150    MOVEUP  DEC VIC+1   ;Decrement the y
        coordinate of the sprite - when it reaches less than
        zero it will flip to 255
2165 ;
2170    HORIZON  LDA DELTAX  ;Now we test the
        horizontal movement of the sprite on the screen
2190    BEQ RIGHT1  ;Joystick unmoved so exit via right1
2200    BMI MOVELEFT ;If deltax = 255 then branch to
        moveleft else drop through to next instruction
2205    ;
2210    MVERIGHT INC VIC     ;Increment x coordinate
2230    BNE RIGHT1  ;We can test if 'vic' is zero or not if it
        is zero then we know we have moved onto the right
        side of screen so Increment Xmsb (if you were using
        more
2240    INC VIC+16  ;sprites this simple method could not
        be used, we will discuss this later)
2245    ;
2250    RIGHT1  RTS ;After dealing with right movement
        return from the routine.
2265;
2270    MOVELEFT DEC VIC     ;All that was said above
        about moving right applies to moving left, with slight
        exceptions
2290    LDA VIC     ;We cannot test for 'vic' equalling zero
        because that would indicate we are still inside the
        right-hand side of the screen so we must transfer
2300    CMP #255    ;'vic' to the accumulator and compare
        it with 255, just inside the left of the screen
2310    BNE LEFT1
2320    DEC VIC+16  ;If it does, clear bit zero of the xmsb
        ($D010).
2330    LEFT1   RTS
2350    ;
```

For our simple example program I decided to make the function of the fire button change the colour of the sprite. Here is the FIRACTN routine to do just that. All very simple.

```
2400    FIREACTN INC DELAY+1  ;We need a further delay
        here to slow the effect of the button down still more
        as before, it the
2420    BNE EXITFIRE ;Delay counter is not zero then skip
        the routine
2430    LDA #192    ;Reset the delay counter ready for the
        next iteration of this routine.
2440    STA DELAY+1
2450    NC VIC+39   ;Just increments the colour register of
        sprite zero so that we can actually see the joystick
        routine working.
2460    EXITFIRE  RTS
```

I suppose the delay section of the 'movesprite' routine could be enhanced by testing the fire button before incrementing the delay counter, then if some action was required by the 'fireactn' routine it could be done immediately. This would obviously depend on what action was taking place - maybe letting loose an energy bolt in a game might require more rapid firebutton response. After BUTTON had been tested, and acted upon or not, it could then increment the delay counter.

It might be less confusing if instead of incrementing the delay counters they were, instead, decremented. But that is where you can experiment. By trying various values in the delay counters you can determine just how fast, or slow, you want things to happen. Take out the delay just to see how fast Machine Language will move a Sprite!

## FINAL WORDS

**The point of this months encounter is to note how a program can be developed using a constructive view - top down planning, rather than a random ad lib approach, by 'bunging' in routines as and when they are thought of. Also, the joystick routine, along with the division routine, can be saved out to your 'Library of Routines' disk, which, hopefully, you will have started with the Multiplication routine.**

Till next time, bye for now...

# MOUSE 80

**Do you own a C128 and a mouse? Then here's the utility to use them together in 80-column text mode - D. H. FABER**

**When I recently purchased a mouse (Commodore 1351), I was not too surprised to find that the accompanying software contained a C128 mousedriver for the 40 column mode only. After all, the 8563 VDC chip (or 8568 in the newer models with 64K video ram) does nopt support sprites, and its complicated access method doesn't help in creating software sprites either!**

However, I intended to use my new acquistion mainly in my own programs for the selection of options (or filenames etc.) from the 80 column screen. If you have similar needs, the utility MOUSE80 described below is the answer. If you don't own a mouse, please read on anyway, because joystick control is also provided!

## HOW IT WORKS

At this point. let's assume you've the machine code utility MOUSE80 and the basic demo program DEMOMOUSE80.BAS at hand. Let's combine an X-ray of the utiliy's inner life with some action on your part! Connect the mouse to either port (if no mouse signals are detected on port 1 then port 2 is tested), or, if you don't have a mouse plug in a joystick instead (if nbo mouse signals are detected on either port, then joystick control is assumed; note that these tests occur within the MOUSE80 utility itself, not in the demo program).

Now load and run DEMOMOUSE80.BAS (tape users beware: the first thing it wants to do is load MOUSE80). If all is well, you'll see a screen with lines numbered from 0 to 27, using various colours for each line (if you - like me - have to make do with a monochrome halftone you'll see one additional halftone only).

On the upper half of the screen, the two character sets are displayed, the lower half shows some options that can be selected. The mouse's cursor takes the shape of an arrow, initially situated somewhere in the lower region of the screen. (MOUSE80 was written for the Commodore 1351 mouse; since I cannot vouch for the compatibility of other brands, you'll have to try them for

yourself).

Feel free to move the arrow around, but don't press any buttons yet! Joystick users will note that the arrow accelerates as the movement continues, its speed is doubled with each of the four available "gears"; with a proportional mouse there's no need for such acceleration.

Notice anything funny on the fourth line? There are six adjacent characters showing fragments of the arrow and the characters it is currently moving over. This has to do with the way the soft-sprite is generated: the cursor (arrow) affects up to six character positions; the bit patterns are taken from the character set, the sprite is overlaid and the six "reserved"
characters are used to display the "sprited" ones.

This process is repeated continuously until one of the mouse buttons 9or the fire button of the joystick) is pressed. The six characters sacrificed for this purpose are rather obscure ones:
REVERSE+CBM+Y/U/O/H/J/L (screen display codes 244-249 in the graphic set).

This is no great loss, especially since the same characters are still available in the alternative set. Anyway, if you don't like the choice you can select a different set of six adjacent characters to be sacrificed for this purpose (see below).

Two more things are noticeable about the arrow's movement. Firstly, it doesn't leave the screen area. In the vicinity of the borders it even changes direction to be able to point at the borders "from within". Secondly, although it·moves pixel-by-pixel in a vertical direction, in horizontal direction it moves tow pixels at a time.

This was done not out of necessity, but for practical reasons only: it reduces the required mouse-movement, which otherwise would be twice as large horizontally as for the 40-column screen. Also, this utility was not designed for graphical packages, so greater accuracy would be superflous.

Now, avoiding the "selection areas" for the time being, try "clinking" on various places on the screen. In the right-hand bottom corner of the screen, you may read at what position exactly you clicked. Here the character positions are counted from zero onwards (as in Basic's CHAR instruction). note that the maximum pix value in horizontal direction is 638 instead of 639, since the arrow points to the leftmost point of a pair of pixels (see above).

What happens is that after pressing a button, control is returned to the calling program, which can then examine the latest cursor position and decide what to do next (if not on an OPTION area, continue with the arrow in the same position).

The calling program can also detect which button (in case a mouse is used) was pressed (see below). This demo program treats both buttons alike.

You may have noticed that the arrow takes on the colour of the characters it is displayed on, unlike the normal cursor which temporarily "paints" the character it is on. There exists another possibility however: try clicking on MONOCHROME. This results in a screen in one colour only but some characters are now looking weird.

What's happened? If the ATR-bit in register 25 of the VDC is turned off, then the foreground colour is taken from the left nibble of register 26 and not from the attributes. However, the processor doesn't look at the attributes for the characterset bit either! Instead, for all characters the graphic set is assumed. Therefore, if you want to use the monchrome mode, build your option screen with characters from the graphic set only (you may consider swapping the sets in video ram to use characters from the other set instead).

One more remark on this: the colour used for the monochrome option is the one used in the most recent PRINT statement as stored in location 241 ($F1). You may also poke the desired colour code directly into this location.

To continue our guided tour, click on COLOUR to restore the multicoloured screen. Up to now we've used a steady arrow. If you feel it's difficult at times to locate its position, especially on a crowded part of the screen, try clicking on BLINK FASTER and BLINK SLOWER. You'll find that you may choose between a steady cursor(speed=0) and one that blinks slowly (speed=15) to quickly (speed=1). This option can also be easily initiated from the calling program when using MOUSE80 (see below).

Before ending the demo by clicking QUIT, some more remarks are in order. The VDC must be in 80-column mode, 8*8 pixels per character (as on power-up). You may change the start addresses of screen, attributes and character sets, and you may also change the number of lines displayed. If you're not sure how to do this, it may help to consult the listing of the demo program.

For the sake of completeness: MOUSE80 is not wedged into the IRQ routine as is usually done with mouse drivers. The reason is to be found in the VDC's complicated access method. Changing or reading the contents of a location in video ram (or even a register) requires a multitude of machine code instructions, and another program trying to access the VDC can easily corrupt some of its contents, with unpredictable results. Therefore MOUSE80 returns control only after a mouse button has been pressed; in the meantime it even inhibits interrupts to avoid keyboard interference.

## IN USE

The machine code file
MOUSE80 loads from $E000 to $E9E0 in RAM 0 (the area from $E9E0 to $EA6F is used as a scratch pad). From BASIC the mousedriver is called as:

**BANK0:SYS57344,A,X,Y**

The Meaning of the parameters A,X and Y (which are transferred to the accu, X-register and Y-register respectively) is as follows:
X = horizontal character position at which the sprite will first appear (0-79).
Y = idem vertical character position. Normal range 0-24 or higher if you changed the number of lines displayed.

Note that the parameters are not checked! A special case is X=Y=255: the sprite will reappear at the same spot it was in when the previoud call to MOUSE80 returned control to the calling program. This is useful to restart the driver if you click on an irrelevant part of the screen.

A = blink speed. 0 for no blink, 1 to 15 for fast to slow blink. If A<16 the mouse driver operates in colour-mode. To use monochrome mode add 128 to the value of A; the colour used is taken from the most recent PRINT statement or may be poked directly to location 241 (=$F1).

Control is returned to the calling program after pressing one of the mouse buttons (or the fire button of a joystick). Relevant information is stored in registers A, X and Y. To obtain these values use:

**RREG A,X,Y**

(A) Is zero if the left mouse button was pressed, 1 for the right button (its value is irrelevant for a joystick). The values of X and Y are the horizontal and vertical character positions respectively at which the arrow was pointing. you may find the exact pixel values as follows:

**BANKO**
**SYS57402:RREG A,X:XPIX = A + 256*X**
**SYS57427:RREG A,X:YPIX = A + 256*X**

If you want to sacrifice different characters to generate the soft sprite, you should;

POKE the screen display code of the first one to location $E9DE (59870); the default value is 244.
POKE the characterset (0 or 1) to location $E9DF (59871); default value is 0 (the graphic set). For ASSEMBLER programmers, you may call the driver from anywhere in RAM 0 below $D000.

On return A,X and Y contain values whose meaning is described above. To obtain the exact pixel location of the arrow use;

```
LDA #$3F
STA $FF00
JSR $E03A
```

to obtain the low and high byte of the X-value in A and X resspectively. (Use JSR $E053 likewise to obtain the Y-value).

If you wish to call these routines from underneath the I/O area ($D000-$E000), or from a different RAM bank or a cartridge or the function ROM, you'll have to use the kernal JSRFAR routine. If you really intend to do this, I expect you will know how it works, if not consult a recently documented ROM listing.

MOUSE80 does of course contain subroutines to access the VDC's registers and the video RAM. As a bonus to assembler programmers, here's how to use them for your own purposes.

Assuming the calling program is in RAM 0 and not underneath the I/O area, you must select a bank with RAM 0 and the I/O components;

```
LDA #$3E
STA $FF00
```

**(from $d000-$E000 or from other banks you must use JSRFAR, see above). You can now use five subroutines as follows;**

**REGWRITE - JSR $E003 (A,X) - value X is stored in register A.**

**REGREAD - JSR $E00F (A) - value of register A is stored in A.**

**VDCSRITE - JSR $E01B (A,X,Y) - value Y is stored in video RAM as A/X (low byte/high byte).**

**VDCREAD - JSR $E025 (A,X) - the value in video RAM (A/X = low/high byte) is transferred to A.**

**SETUPDATE - JSR $E02D (A,X) - the contents of A and X are transferred to registers 19 and 18 (UPDATE low and high respectively).**

To Basic programmers, these subroutines are not available since there exists no BANK command to select the required memory configuration. However, if you consult the listing of the demo program and copy the DATA statements and the lines POKEing them into memory, you'll have the same facilities available in BANK 15 by;

# SPRITE EDITOR

**Any serious games programmer needs a sprite editor to aid design. This program should make working with sprites easier and more effective - BRIAN RHODES**

Serious programmers are always looking for new utilities to enable them to improve on their finished product. A sprite editor is an invaluable asset when it comes to designing and manipulating sprites. Here we present an editor which we hope will answer all your needs.

## EDITOR INSTRUCTIONS

| | |
|---|---|
| F | - Function. |
| SH | - Shifted. |
| M/C | - Multi-colour mode. |
| CRSR Keys | - Move cursor. |
| SPACES | ots all colours. |
| 1 | - Plots sprite colour regardless of m/c mode. |
| 2 | - Plots m/c #0 in m/c mode only. |
| 3 | - Plots m/c #1 in m/c mode only. |
| DEL | - Unplots all colours (Any mode). |

**The shifted version of the above also work so shift lock can be used.**

| | |
|---|---|
| + | - Next sprite. |
| - | - Previous sprite. |
| 0 | - Delay before + or -, (used in animation). |
| + | - Increases delay. |
| - | - Reduces delay. |
| (Pound sign) | - Jump forward 10 sprites. |
| SHIFT (£) | - Jump back 10 sprites. |
| P | - Progress (copies present sprite to next location and advances). |
| A | - Auto advance CRSR. |
| CLR SHIFTED | - Erase sprite. |
| HOME | - Home CRSR. |
| R | - Restart (return to menu). |
| Q | - Quit (return to Basic.SYS49152/24576 RETURN restarts). |
| @ | - Scroll mode. use CRSR keys to position sprite. SPACE exits. |
| F | - Fill sprite. |
| | - Flip (reverse around horizontal axis). |
| | - Mirror (reverse around vertical axis). |
| = | - Equalise, because of the nature of spritesmirroring ( ) m/c sprites spoils the colours. Pressing "=" restores the correct colours. |
| C | - Copy sprite. |
| F7 | - Toggle between m/c and normal mode. |
| * | - Change colours. |
| F1 | - Background/border colour. |
| F3 | - Sprite colour. |
| F5 | - m/c#0 (only in m/c mode). |
| F7 | - m/c#1 (only in m/c mode). Also see note in multi sprite mode. |
| Shifted F | -keys step backwards. |
| D | - Data output. |
| F1 | - Disk storage. |
| F3 | - Printout. |
| F5 | - Numerical Printout. |
| F8 | - Sprite printout (normal mode only). |
| K | - Kill sprite advance. Toggles modes A,B and off. |
| OFF | - All sprites follow grid. |
| A | - Only one sprite follows grid. |
| B | - No sprites follow grid. |
| M | - Multi-sprite mode. Enables objects of several sprites to be constructed. |
| SPACES | - Exit mode. |
| CRSR KEYS | - Move sprite. |
| NUMBERS 1-7 | - Change controlled sprite. |
| F2 | - Enables sprites 5,6 & 7. |
| F7 | - Toggle m/c and normal mode. |
| + | - Next sprite. |
| - | - Previous sprite. |
| < | - Increment colour. |
| > | - Decrement colour. |
| (Up arrow) | - Expand vertically. |
| (Left arrow) | - Expand horizontally. |

### NOTES

1) All numbers wrap-round i.e. when they get to 255 they revert to 0 (and the other way, or at 15 when using colours.

2) Entering null values (0 or "") normally exits present mode.

3) When using multi sprites be careful that colour change and mode change do not spoil multi sprite parameters.

## ADDITIONAL NOTES

These are some additional notes on the use of the Sprite Editor. They are meant to supplement and expand upon the instructions and explain in detail all available functions.

There are two forms of the sprite editor provided, each occupying a different area of memory.
1) Entitled "SPRITE ED.$6000" (29 blocks), this is a single file version loading form $6000 to $9400 and is run by the command "SYS 24576 <RETURN> ($6000).
2) Entitled "CODE 1 C000-D000" (17 blocks) and "CODE 2 6800-7C00" (13 blocks), this is a two file version loading between $6800-$7C00 and $C000-$D000 and is run by the command "SYS 49152 <RETURN> "($C000).

## PARAMETER SCREEN

When the Program is run, the first thing that will be seen is the parameter screen. The screen will clear and the words "PARAMETER INPUT" will appear at the top of the screen along with a copyright message. You will then be asked to input a number of variables such as Colour, Start Sprite etc. and a cursor will appear beside the present value, if this value is correct then just press Return, if not then use delete and the numerical keys to change it. The questions that will be asked are as follows:
1 START SPRITE - This is the sprite that will first appear on the screen. Sprite numbers are identical to the number POKEd to 2048 to display sprites normally. It is inadvisable to use sprites below 128 but for more information see the Commodore manual.

2 BACKGROUND COLOUR - This is the colour of the background on which the sprites will be displayed.

3 NORMAL MODE COLOUR 1 - This is the colour of any hi-res sprite.

4 M/C MODE COLOUR 1 - This is the colour that will be given to any multi-coloured sprite as its basic colour.
M/COLOUR #0 - This is the colour that will be given to m/c # 0 of any multi-coloured sprite.

M/COLOUR # 1 - Likewise for m/c 1.

5 Advance/Retreat Interval - This determines the number of sprites that will be advanced or retreated for every press of + or -.

6 HIGHEST SPRITE BEFORE WRAPROUND - When on this sprite press + and the program will go to the Lowest sprite.

7 LOWEST SPRITE BEFORE WRAPROUND - When on this sprite, Press - and program will go to the Highest sprite.

If 5, 6 and 7 are given such values as to be impossible the program will return to question 5.

## EDITING SCREEN

When all six variables have been defined the editing screen will appear. This can be split into three sections.
1. The editing grid.
2. The sprite display area.
3. The control/input line.

The editing grid is a block of squares on the left of the screen which displays an enlarged version of the current sprite. A flashing cursor will also appear on the grid and this indicates the position of any change to the sprite.
To the right of this is a space coloured according to the background colour. It is on this space that the defined sprites appear.
The top line and the space above the display area is the control/input area where instructions or requests will appear.
The standard editing commands are given in the instructions. Most of the commands are straight-forward but for the more complicated ones a more detailed description is given below.
0 - This controls the time interval between changing the sprite number and control being returned to the keyboard. This is useful in animating sprites when the upper and lower sprite wraproud limits can be set to the limits of the animated sprite and + or - is held down to cycle through the images. Varying the delay value changes the speed of the animation. 0 is the minimum delay, 255 the maximum.
A - Cursor auto advance. This allows a preset cursor movement after any point has been plotted. When selected a 9 point grid will appear and below this is the present value. The keys 0 to 8 now select the direction of the advance. Any UNPLOT command will act in the opposite direction. A value of 0 turns auto advance off.
F8 - Sprite print-out. Pressing this key will print out the currently edited sprite on an MPS803 printer. The printout will be in normal hi-res mode only.
D - Data output. Pressing this gives access to a sub menu.
Q - Quit mode.
F1 - Data storage on disk. This gives access to a standard Load/Save facility.
F3 - Display. This gives a numerical screen display of the data that goes to make up a particular sprite.
F5 - Numerical print-out. This is the same as above but to printer (MPS803).

## MULTI SPRITE MODE

It is possible using the editor to create images using more than one sprite. To do this the sprite advance mechanism must be turned off. This prevents the sprites from changing when + or - is pressed and is achieved by pressing K to enter "Kill mode". An A will appear at the top of the screen. Press K again and B will appear. Pressing it a third time will make the latter disappear.
Kill mode A means that only sprite number one will follow the main grid and in kill mode B no sprites will follow the grid.
The number keys 1-7 change between sprites and the cursor keys move them. SPACE turns the sprite on or off and UP ARROW and BACK ARROW expand the sprite in the x and y directions. (,) and (.) change the sprite colours whilst + and - change sprite numbers and F7 changes between m/c and normal mode.
Normally only four sprites are visible but pressing F2 brings the other sprites on. Use F2 once only because pressing it again will reset the positions of sprites 5, 6 and 7.
N.B. When using multi-sprite mode, be very careful about changing mode and colours of the sprites since it will affect all sprites and may destroy the layout.

# SMOOTH VERTICAL SCROLLER

**Is your vertical scrolling just a little jerky? Use this program and stop buying the aspirins - Suman Roy**

**There must be many computer users who have video recorders, and would like to use their 64 or 128 to create video titles for home productions, etc. One way is to use the standard size Commodore text, but this tends to look insignificant on a video title screen. With this program, you can create titles that appear in double size characters (twice the height AND width of standard text) with the addition of smooth vertical scrolling. Even if you don't have a video recorder, the program is great fun at parties (what sort of parties do you throw? -Ed.), and can be very useful at meetings.**

## HOW TO USE

**To use the system:**

1. **Load in the code either via the CDU MENU or directly.**
2. **Type NEW**
3. **Load the editor (Part 2), and run it.**

The screen will go white, and a black square should appear in the upper left corner of the screen. This is your cursor. Messages may be typed in, but only up to 16 characters may be entered per line. The cursor will not allow you to advance past column 16 of the screen.

After typing in each line, press RETURN. This is very important as each line is processed only after the RETURN key is pressed. This editor is a pseudo-screen editor, that is, it behaves rather like the screen editor you use to type in programs on the 64. However, only alphanumerics may be entered, and, only in upper case. In addition, the ".", and "-" keys may be used, not to mention spaces.

The cursor up/down keys do what you might expect, allowing you to type in messages down all 25 rows of the screen. Cursor right also functions as normal, but use the "del" key to perform a cursor left. With a little practice, you will be able to enter messages quickly and efficiently.

This system can be used to produce hundreds of lines of titles in one go, and so I have allowed the screen to scroll both up and down to enter messages in the correct places.

One point to note is that the routine will automatically centre the text on screen when it comes to output the finished product, so do not precede messages by leading spaces.

If you use the cursor keys to edit a message, remember to press return afterwards to reprocess the modified line. Before pressing return, though, move the cursor to the end of the line, as otherwise the character that the cursor is on and all characters afterwards on the same line will be deleted.

When you have entered your messages, you can view the title by pressing the asterisk key.

# SPRITE PRIORITIES

## Make use of the Priority properties of your sprites - J. SIMPSON

**If you want to achieve a truly three-dimensional quality to your sprites here is an incredibly useful routine for you to use. It's very short, so does not take up too much disk space. Examples of three-dimensional sprites which spring to mind are the various sport simulations, such as "INTERNATIONAL SOCCER".**

### ABOUT SPRITES

As you probably know, sprites have a definite priority arrangement in that the lower the sprite number the higher is its priority. This means that sprite 0 has the highest, through to sprite 7 which has the lowest.

Sprites with higher priority always display in front of those with a lower priority. This is fixed within the hardware of the VIC II chip, which means if you want to create a three-dimensional illusion, then a routine needs to be constructed which will manage the sprites by keeping "foreground" sprites higher in priority.

To handle this, I decided that sprites lower down the screen would be considered as "foreground" to those higher up the screen. I'm sure you know that the pixel map is arranged with the "Y" co-ordinate running from 0 at the top of the screen, to 255 at the bottom. The Machine Language routine, SPRITE PRIORITIES, uses the "Y" co-ordinate information to decide which sprite should be where; the sprite with the greater "Y" value becoming sprite 0, and so on.

### WITHIN THE ROUTINES

1 SPRITE PRIORITIES this is the ML routine which deals with checking and updating all the sprite positions and their priorities. If a sprite moves up the screen (or backwards into the picture), and in doing so it passes above (or behind) another sprite, then SPRITE PRIORITIES will swap the two sprites around, together with all data relevant to each (ie: Image, Mcm, Xpos, Ypos, Colours etc). This neatly holds together the illusion that a sprite's priority is changing - first passing in front of, then behind another sprite. This does, however, pose a tricky problem! Let's say that the joystick is being used to control one particular sprite element. Normally that control would be defined and controlled using one particular sprite - say, sprite 0. All that needs to be done is to peek and poke (or ML equivalent) with "Y" co-ordinate information into sprite 0. However, should sprite 0 move up the screen and pass the next sprite, then CONTROL shifts to sprite

1, and if CONTROL carries on up the screen it might become sprite 2, 3, 4, 5, 6 or 7.

2 BASIC DEMO - For programmers, and to show the system working, the basic demo program outlines a demonstration of SPRITE PRIORITIES in action, as well as useful routines for the controlling of which sprites are where. When you "RUN" the program (that is after loading SPRITE PRIORITIES, and typing "NEW", then loading BASIC DEMO), eight sprites are displayed diagonally across the screen. The images are simple coloured blocks. Their default values are:

**Block 1 = White**
**Block 2 = Red**
**Block 3 = Cyan**
**Block 4 = Purple**
**Block 5 = Green**
**Block 6 = Blue**
**Block 7 = Yellow**

### Block 8 = Orange

You can use the numerical keys (1-8) to select any one of the eight blocks. That block will now be under CONTROL, and can be manipulated around the screen using the joystick plugged into port 2.
If you study the listing, you will see that in the INITIALISE routine (lines 68-78) variables "N" and "K" have been declared, and the two arrays F() and DS() dimensioned. At line 76 the arrays are filled - F() with 0 to 7 and DS() with the "Y" co-ordinate value of each sprite, from 0 to 7.

The array F() holds the current position of the BLOCK (not the sprite) on the screen, the DS() - which derives from "Dummy sprite" - holds the current "Y" location of where each sprite would be, should there be no SPRITE PRIORITIES manipulation.

The variable "K" is used to shift joystick CONTROL over the block selected from the numerical key input. A loop checks through the F() array to find the current screen position in relationship to the sprite. For example, BLOCK 5 might be at the bottom of the screen, and so it would be sprite 0. Where "N" is set to equal the actual sprite value - in the foregoing example, 0. When the joystick is moved up or down, program control will call either JOYSTICK UP (commencing at line 32). Let's say "up" is the selection. First DS(K) is decremented (K=Block being moved), and the screen parameters are checked (line 22). Then the updated value in DS(K) is poked into the "Y" register of the sprite holding the data for that BLOCK: "N" holds the sprite number.

Next, the CONTROL sprite's "Y" co-ordinate is checked against the next lesser prioritised sprite "Y" co-ordinate. If CONTROL is greater, then it maintains priority and so the program skips lines 25-27 and returns to MAINLOOP - no more needing to be done. However, should the co-ordinate value now be less, then line 25 calls the ML routine SPRITE PRIORITIES, where priority and all relevant sprite data is toggled from one to the other. On return from the ML routine, "N" is now incremented to the next highest sprite number. This is followed by an error trap, and UPDATE F(lag) ARRAY (line 64), which will update F(N) for the current sprite position. Moving down the screen is the reverse of the above - check out lines 32-39. Left and right have no effect upon priority, and so these are standard routines to Peek and Poke "X" co-ordinates.

## A STAGE FURTHER

To take things a stage further and have multiple sprite movement on the screen, the variable "N" would need to become an eight element array. Each image, block, or whatever, would be given a constant value from 0 to 7, and when the program updates "Y" co-ordinates for each element, N (Element Number) would be used. This could be followed by a line such as:

**ON (ELEMENT NUMBER)**
**GOSUB (PARAMETERS)**

Here subroutines would handle differing images, or elements, and/or situations.
I have not incorporated a collision detect routine within the Basic demo as the main purpose is to show how effective SPRITE PRIORITIES is, and to offer a demonstration of a method of control. Collision detect should operate quite normal without any problems. For ML programmers, the conversion of the Basic demo routines into source should prove to be quite elementary.

## SPRITE IMAGE DATA

I have included a short program of data for sprite images which will display, numbered (1 to 8), three-dimensional, coloured boxes. The numbers corresponding with numerical keys and the colours as before. You will now have to make some changes to the BASIC DEMO program.
First, delete line 70 entirely.
Change line 71 to read:

**POKEP + C, PO + C**
**Remove the REM from line 72 and type:**

**POKEV + 28, 255**
**Remove the REM from line 73 and type:**

**POKEV + 37, 11**
**Remove the REM from line 74 and type:**

**POKEV + 38, 12**

This enables the multicolour mode and sets the colours to Grey 1 and Grey 2.

## RUNNING THE DEMO

First load and run SPRITE PRIORITIES then type "NEW". If you are using sprite image data then load this is and type "NEW". Now you can load and RUN the BASIC DEMO program. Remember, keys 1 to 8 will select the BLOCK you wish to control with a joystick plugged into port 2. Well, that's about it. I'm off to the pub for a pint. Hope you like SPRITE PRIORITIES, and can utilise it and expand upon it...

# Sacram

## We continue STUART ALLEN's insight into a future world

**.....Alan now decided to grow.....**

He grew fairly fast, and took the robbers by surprise. Once he had finished (and he was now about eight feet high) growing. He said, in a butch voice, "I am. If you want anything out of this place, then you have got three options. Either you pay for what you want, or you leave, or you come through me. Take your pick.... Oh yes, I should also tell you, that for you ten people, there is absolutely no way out. The doors are hermetically sealed, the windows are bullet proof, well, come to think of it, I am almost bullet proof, as well. What's your decision" "I think ..... We'll leave. If that's ok with you?" "No it isn't, but you can try to leave, if you want to.", said Alan, in the most fearful way. The robbers then tried to get out through the main doors, and sure enough, they were somehow sealed. They then tried the other doors that led into the complex, but alas, they also were sealed. They had a quick think, then decided to go to war. They rapidly went for the weapons stands.

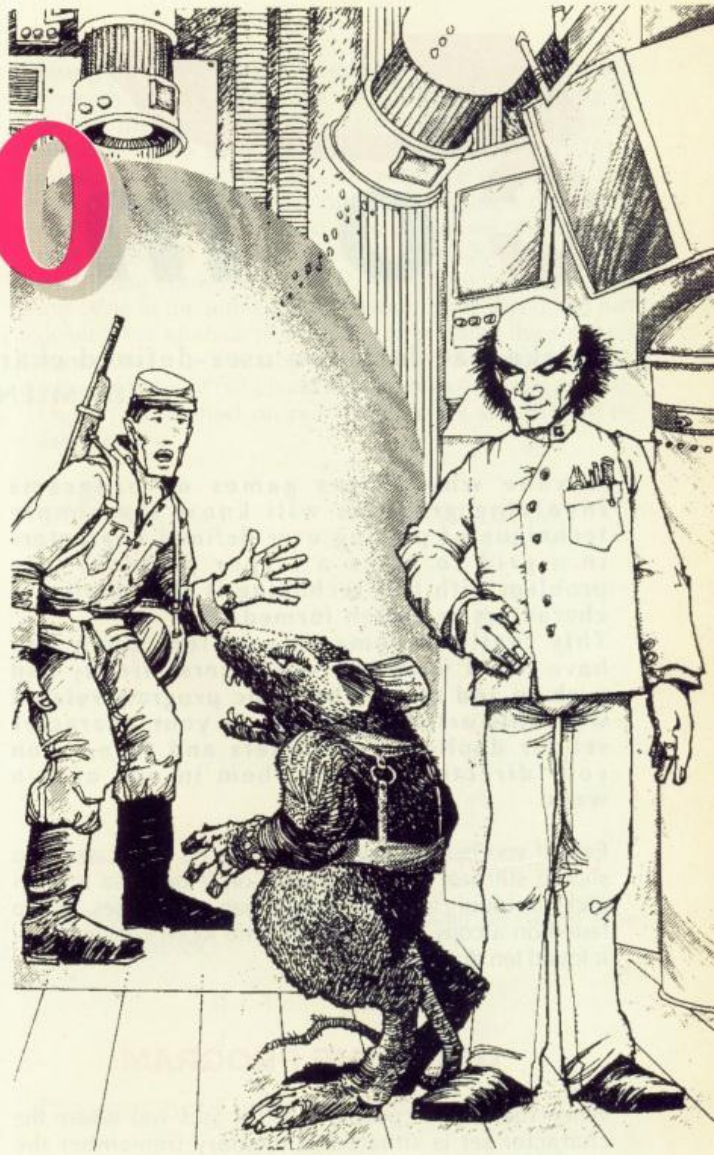
As they reached out to take a weapon of their choice, an electrical field started to glow around them. They still tried to collect the weapons, and ended up, not with the weapons, but with burnt hands. The leader said to Alan, "All right, which option do you want to take?" "The third one..." He then leapt into action, grabbing a gun, after turning off the field, and fired ten shots. He hit each person that wasn't on his staff, and wounded all ten of the assailants. He then said, "Now you know how good I am with a gun, Now feel the wrath of my fists" Just after he finished saying that, he thought, 'boy, did that sound stupid or what?'. He chose one person at random, and gave him just one hit with his fist. Although Alan did not know it, the person that he had selected, was none other than the ring leader. He was also of a weak disposition, so after the hit, he curled up in a pool of blood, and collapsed. Alan said to the other nine, "Who's next? I don't want to sound like a big bully, but I don't like people who pick on others who are much smaller than themselves. It is only when a person, such as I, who you picked on, has got some sort of a magical power to grow, when I like it" He then decided to go for the apparent leader, who was now panicking and screaming for mercy. As Alan advanced onto this person, he was begging and pleading for his life to be spared. Alan said, "Tell you what... I'll only hit you once, or twice... or seventeen times." He then decided to go for the big one, seventeen. The disappointment came when the body literally disintegrated after the fifth punch. He said to the rest, "If you want to leave now, then feel free to. The doors have been opened, and you are no longer my enemies. If you ever try to rob me again, though, then you will not survive to tell the tale." After that, they all fled outside, never to be seen again by Alan, again. He told the staff, "If any of them try to get at you, or you want some building up, then don't hesitate to ask me. I'll do my best, although you won't turn out as strong as me, you will still be strong enough to behave normally, if they, or anyone else attack you" After that speech, the staff all jumped at the chance, so they were going to be trained the next day. He dismissed the staff, telling them, "I will not charge for the service, but if you want to make a donation, then it will be accepted. Those who do, shall get the money put away to help them, perhaps gain in knowledge." They then bid there farewells, and left him for the night. He then went upstairs to awaken the Doctor. As he opened the bedroom, he thought, 'aah. Sweet as a baby. It would be a shame to take him out of his sweet slumber. Still, a mutated Prairie Dog must do what a mutated Prairie Dog must do, therefore Alan decided to wake up the Doctor. After a few moans and groans, the Doctor finally got himself dressed, and went to Alan. He said, "So. You are a man of your word, you said that you would awaken me from my sleep at about five o clock, and what time is it now? It's bang on five." "Well, if I say that i'll do something, then i'll do it. Shall we get going?" "If it's Ok with you", agreed the Doctor. And so, down they went into the garage. Alan said, "Just how big is this machine? If it's big, then I'll have to get out the special vehicle." "If I tell you this, then you shall get a good idea as to the size. I can get about a quarter of it in a sixteen-wheeler." "In that case, it is a job for the special. Wait a minute, I've just remembered the compliment that you gave me when we opened.

You said that what I'd just done, was the tenth wonder of the world. What are the eighth and ninth?" "Well, to me, the eighth is someone being able to have super- powers, like flight, x-ray vision, etc., and the ninth is Time travel. Satisfied", queried the Doctor. "Perfectly. Just for the record, did you know that I can fly, shrink and grow at will, have got x-ray vision, and can travel in time?" "No, this certainly means that you are a bloke to be reckoned with." "True. Now, shall we get going?" With this, Alan got out a small remote control. He flicked a few switches, pushed a few buttons and twiddled a couple of knobs. The pair of them both saw the garage door open, and a

# ento

back wall gradually start to move upwards. It was a slow movement, and they were standing for a couple of minutes. The Doctor was reassured that it was usually faster, but it hadn't been in use for quite some time. When it finally reached the top, out rumbled a sixteen-wheeler, followed by three others. They were all controlled by drones in the cockpit. Alan told the doctor, "This is the special. As many sixteen- wheelers 'strung' together by invisible links, and all piloted by drones. The drones are also the command centre of the machine, receiving any commands from the remote control. To stop unauthorised people from using it, you have to enter the correct number within ten seconds of you operating, because if you do not, it won't react to any commands that you give it. It would also tell me by telepathy if anyone was using it. Shall we get into my car?" "I think that would be appropriate" For anyone watching the side of the building, it was truly in impressive sight. Watching a small car, followed by four sixteen-wheelers, all following the exact same path. If anyone went up to the tyre tracks, they would find that their would only be one set. The journey was relatively quiet, until they reached Chowchilla.

It was there where they received their first piece of trouble. They saw a moderately large jeep in the middle of the road. Alan thought that it would move out of the way, because it obviously had a driver. When things started to get a little bit frantic, Alan flicked a switch on his cars dashboard and on the remote control. This put up an impregnable force field around the convoy, and he still continued driving. He slowed down a little bit, and pressed his horn, though very little of this was actually getting through to the driver. Alan thought, 'I bet the owner of that vehicle is a road hog. If he or she is, then he or she had better take a quick look outside, as it will be the last glimpse of sunlight that they will see'. There was a blinding flash, with bits of ash floating down. The doctor asked Alan what had happened. Alan only said, "The driver has gone to a far, far better place than the one that we are on." That was the only trouble they had until they reached Sacramento. After a long car journey, the doctor suggested that they went into a pub type establishment, to order some liquid refreshment from the bar-tender. The couple were just casually sitting around, talking to each other, when Alan noticed a rather odd-looking pair of mutant animals enter. One, a grizzled old Mule, is unusual only in that the cut of his clothes is slightly different. The other, a good- looking, mutant, Prairie Dog, would stand out in any crowd. He was wearing a bright pink shirt embroidered with silver thread, a red ascot, red trousers, a shiny black jacket, matching black boots, a silver studded belt and holster, a



pearl-handled revolver, and a weird, wide-brimmed hat with little round tassels. There is a shocked silence in the room. Judging by the faces of the other people there, none of them had seen anything remotely like them before.

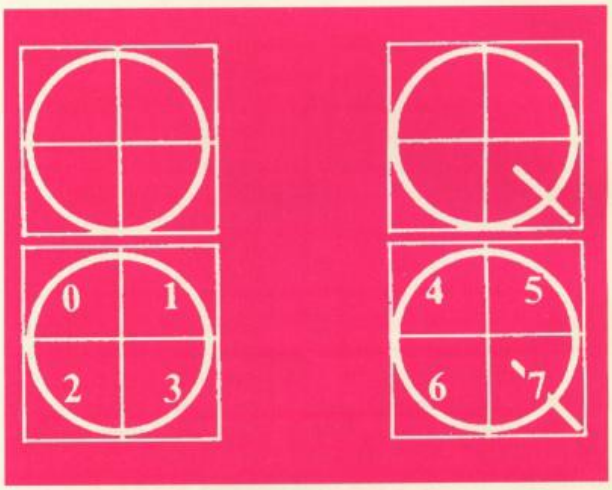
Although amazed by the other Prairie Dog's appearance, Alan thought that he should, at least, introduce himself. He boldly said, "Hi. What are you doing here?" The gaudily dressed, young Prairie Dog speaks, "My friend Jose and I, Don Lazlo Fuego Hoarez de Zapata, come to this land seeking brave drivers and powerful vehicles. The fighters we are looking for must be true of heart and of mind, willing to risk all for a noble cause and a fair lady! Could it be you that we seek?" Alan thought, ' True of heart and of mind? For a fair lady? What kind a guy speaks this garbage. Even though it suits me to a tee. Us Prairie Dogs must stick together, so I shall humour him'. He then said aloud, "I am of true mind and heart, I shall accompany you." Lazlo seats himself and starts speaking, "I come from a far- off land known as West Texas.

# UP Compres

**Make way for more user-defined characters by wiping out the dulplicates -**
**DAMIEN MARSH**

**Anyone who writes games or programs involving graphics will know the simple technique of placing user-defined characters in a grid to make a larger picture. One problem with this technique is that duplicate characters are often formed.**
**This really becomes a problem when you have filled your 256 characters already and wish to add more. The Basic program related with this article will search your character set for duplicate characters and then, upon your direction, delete them in one of two ways.**

Even if you believe that your set has no duplicates, you should still run the compressor on it because it often picks up duplicates that the human eye misses. When tested on a copy of the Commodore ROM character set, it found ten duplicates.




| CHARACTER NUMBER | IS IDENTICAL TO |
|---|---|
| 0 | ***** |
| 1 | ***** |
| 2 | ***** |
| 3 | ***** |
| 4 | 0 |
| 5 | 1 |
| 6 | 2 |
| 7 | ***** |

Table 1 - Duplicate table for Diagram 1

## USING THE PROGRAM

When running the program, it first asks you where the character set is situated in memory (remember the character set should be loaded before the COMPRESSOR). It cannot be stored in any area of shadow RAM below 6144 ($1800) or the program won't recognise it.The program then asks for the last character in the set. Once you enter this, the search will begin for duplicated characters.

This search may take quite a while, as every character in your set must be checked against almost all of the others.Once the search is complete, the program will have formulated a table of duplicates.

This table tells you which characters are duplicates of any of the others and could therefore be removed. If, for example, our character set contains eight characters (zero to seven) which form two grids to make up the letters O and Q as shown in Diagram 1.

The chances are that characters four, five and six will be duplicates of zero, one and two, respectively. A table of duplicates would look like Table 1.

A character with five stars in the right-hand column is an original whilst the others are duplicates of the character whose number is displayed. There are options to list this table to the screen or printer. When listing to the screen, a key must be pressed after each screenful of information.

You may now quit the program, delete the duplicate characters, and, if necessary, compress the set. if you choose to delete the duplicate characters, you will be asked for a value with which to fill them. After the

# Gsor

operation is finished, a conversion table will have been created which gives all the information that you need to convert the characters in your old character set to their respective characters in the new set. A conversion table for our example characters would look like Table 2.

| OLD CHARSET | NEW CHARSET |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 0 |
| 5 | 1 |
| 6 | 2 |
| 7 | 7 |

**Table 2 - conversion table**

Make sure you always keep a copy of the characters used to make up your grids when using this program. First draw a grid for each character block as in Diagram 2(a).

```
0 1     4 5
2 3     6 7
```

**Diagram 2(a) Original character grid 2(a)**

Then draw another grid set ready to put the new characters in. Look down the column on the left for the old character number and read across to find it's new value in the right-hand column. Now write this new number in the correct place on the blank grid. Repeat this process until all your grids are filled. Our example character grids should now look like Diagram 2(b).

```
0 1     0 1
2 3     2 7
```

**Diagram 2(b) Deleted character grid 2(b)**

As you can see, characters four, five and six are now unused. If you were now to load the new set on your character editor, you would see that the deleted characters are now filled with the code you entered earlier in the Compressor program. You could now fill these characters with something else without causing damage to your grids.

Using this method gives you free characters scattered throughout your character set. Instead of choosing to delete the duplicate characters, you may choose to remove them and compress the set. This method compacts the original characters down over the unwanted duplicates, leaving all of the blank characters at the end of the set. If you have a large set, this process may take a few minutes.Once again you have the option of listing a conversion table to the screen or printer. Use this table in the same way you would have if you had just deleted the characters without compressing the set. You may notice, in the table created by this process, that some of the original character's numbers change as well. Using this method on our example set would give us Table 3.

| OLD CHARSET | NEW CHARSET |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 1 |
| 5 | 2 |
| 6 | 3 |
| 7 | 4 |

**Table 3 - Conversion table after compression**

The character grids for the new character set would look like Diagram 2(c)

```
0 1     0 1
2 3     2 4
```

**Diagram 2(c) Compressed character grid 2(c)**

Whichever method you use, it is very important that you have a hard copy of the conversion table. If you do not own a printer, you should list this table to the screen and copy it by hand. This may seem like a lot of work but it is worth it in the long run.The serious games programmer or graphics artist will find this utility very useful indeed. To test the program, I created a character set comprising of the letters A to G, each in a three by three grid. The original set used over sixty characters but it compressed down to less than thirty.

## FINAL NOTE

**There may be cases where you might deliberately wish to retain a set of duplicates such as the letter O with the number zero. In this case, you will have to change one of these before using UDG Compressor and remember to change it back afterwards.**

# TECHNO-INFO

**Due to the technical problems of last issue, we bring you a bumper edition of TECHNO-INFO to make up for the missing one - JASON FINCH**

Welcome to this issue's dose of Techno-Info letters. There were no letters in the July issue, due entirely to circumstances beyond the control of the Editorial staff, and only five in the last issue. Because so many people lost out on those occasions there is a rather large pile of letters mounting up ready to be answered - over sixty in fact. Of course, it is not possible to publish that many all in one go, but we shall try and get through a fair number of them. Therefore, this month, I am very pleased to present to you a BUMPER TECHNO-INFO SECTION that contains not only a staggering TWENTY-FIVE letters and their replies, but also the regular UPDATE and TIP OF THE MONTH sections as well. And as if that wasn't enough for you to be getting on with, the time has come around for yet another TECHNO-INFO CHALLENGE. Details of that follow towards the end of this marathon section. I guarantee that there will be something of interest for everyone this month, and we at CDU trust that you will approve of such this larger-than-life letters section, so without using up further valuable space, let's begin the first query.

## AMIGA 64?

Dear CDU,
In a previous edition of CDU I read that another reader is looking for a "true descender" chip for a Commodore MPS801 printer. I have the Printer IV (Datel) chip which prints descenders. I am willing to sell it for £10 if anyone is interested. That's the business side of my letter out of the way, now for the niceties - thanks for a great mag, keep it up. Now for the question - I have obtained an AMIGA 500 but because I have a lot of software for the 64 I intend retaining it (how else can I use those wonderful disks from CDU) - How can I use some of my Commodore 64 software on my Amiga? Is that what an EMULATOR is for? PLEASE would you supply me with details.
**V.Perry, Barry.**

Dear Mr.Perry,
If anyone does want the chip, I will be sure to pass their address on to you. An emulator is indeed what you require, and for more details about obtaining one you should ring Radical Shareware on 0502-517362 and ask to talk to a guy called Paul. John Simpson, he of numerous lengthy articles, has assured me that Paul will be able to help you. I just think it's a shame that you can't get an Amiga 500 Emulator for the Commodore 64!

## C128 PD & LEAD

Dear CDU,
I have recently become the proud owner of a C128 after having a C64 for four years. I do however have a few queries which I hope you can help me with. First, I am interested in using the CP/M mode, and although I have the System Disk, I have no other software to use with it. Could you tell me where I can get some CP/M programs and PD software from? Also, is there a lead available that would allow me to use 80 columns with my TV? If not, which monitor would you recommend me buying? Could you tell me where I can get some general Public Domain software from for use on the C128? Keep up the good work and C128 support.
**Mark Hopwood, Chester.**

Dear Mark,
To deal with your second query first - it is not possible to connect your computer to a normal television set and have an 80 column display. Unfortunately no hardware has been produced to perform such a task. You will therefore need to buy a monitor. Any monitor with a composite video connection will suffice because then you can buy a converter lead. However, if you would consider paying a little bit more for a better all round monitor then I would suggest that you buy the Commodore 1084 monitor which has not only a composite video connection for 40 columns but also the proper RGB input connection allowing direct connection to the C128 port. You may find that the monitor has various suffixes on the "1084" bit. The C1084S monitor for example just means that the monitor outputs stereo sound - this is used mainly with Amigas though. With regard to your PD query, there are various sources. One that you may try is Kingsway Computers Services, 72 Glencoe Road, Sheffield, S2 2SR (Tel: 0742-750623), their disks being £2.95 each. Alternatively, and cheaper, you could try FSSL, Masons Ryde, Defford Road, Pershore, Worcestershire, WR10 1AZ (Tel: 0386-553153), their disks being only 95p each. They have just introduced four new CP/M Public Domain disks which is

obviously what you are after. They also have various PD software for both the 64 and 128 (in 128 mode of course). Perhaps you could write to or ring up both companies and ask for a catalogue of PD software and then take it from there. I hope you find all this information of some use.

## CHALLENGES ABROAD

Dear CDU,
We are reaching the end of June and I have not yet received my magazine. In fact I received the May issue last week. As you suggested (and I thank you for that) I should have written to you and let you know but I couldn't because of some exams that I'm taking. I don't expect you, of course, to send me now this month's issue as there is no time left. I just want to ask you when the next Challenge will be and if it would cause any difficulties to you or to the magazine if you had to send me every T.I.Challenge by post.
**George Thalassinos, Athens, Greece.**

Dear George,
I appreciate the fact that it can take up to a further two months or so for readers that live outside the UK to receive their copies and I have no quarrels about sending details of the Challenges to you through the post. However, if one person wants this star treatment, everyone will start to want it as I am sure you will appreciate. Therefore, rather than setting the closing date before you have even had opportunity to get hold of the magazine I am extending the closing date for receiving entries from abroad by three months. I think that this will prove to be the fairest method. For people in Britain there is, I'm afraid, still only a month to get things done though! Thanks for showing such interest in the Challenges - remember that the closing date for this one will not be until the end of the year for you, George, and other readers that live outside the British Isles.

## D-A IN THE CLUBHOUSE

Dear CDU,
It was only when the magazine closed down that I realised just how important it was. I felt cut off from fellow enthusiasts and the knowledge gained burning the midnight oil had now become redundant. So; now we have our club back, and if CDU be the club, then surely "Techno-Info" is the club-house where members meet to exchange problems and ideas. The 64 is such a clever little beast that to become an expert in all areas is asking too much, hence the importance of the club. Somebody somewhere knows the answer to your problem. My own problem concerns direct access using code. Specifically how does one translate 'PRINT#15,"U1";Ch;0;Trk;Sct' or B-P, etc. In BASIC these commands require two channels to be opened: a data channel and a command channel. So I began with the KERNAL SETLFS, SETNAM and OPEN routines, but I could not find the correct syntax to send the hash sign (#). Next I tried the LISTEN routine but I got stuck trying to separate the two files, so I spent hours leafing through back issues of Your Commodore going back to 1985, and I found a dozen articles on doing it in BASIC but not a hint of how to translate it into machine language. Could you write a few lines to demonstrate how to do this as I am truly bogged down and I have run out of ideas. Best wishes for the future.
**Roger Harris, Birmingham.**

Dear Roger,
It is pleasing to know that you find the magazine and this section in particular very valuable. Converting Direct Access commands into machine language is quite tricky on first attempts as you seem to have found out. To explain the methods here, though, would take up an awful lot of space and would probably only serve to add to the confusion. But don't despair - on this issue's disk you will find a program titled "CLUBHOUSE D-A" which is an assembly language listing, fully commented, that is sure to be of help to you. I trust that you will find it of some assistance, as will all the other readers that are having difficulty transferring this advanced aspect of drives from BASIC into machine language.

## DIGITAL LAB

Dear CDU,
I wonder, could you do me a favour and ask your readers if anyone has an unwanted copy of a programme called "Digital Lab" by R2RD (whatever that means). It used to be available way back in 1985 from Associated Services (London) Ltd, but they went out of existence some time ago. This is a cry for help as I have tried many other places to get a copy as it would help with my everyday work. It's a great pity that CCI stopped printing, as with that magazine as well as yours being a wealth of information (I have every copy of both CCI and CDU) it used to place adverts. Perhaps you could start a similar thing for CDU, as CDU is now the only decent 64 magazine for serious 64 users. Keep up the good work and long may you continue.
C.D.Roberts, Cornwall.

Dear Mr.Roberts,
By printing your letter, your plea has been granted. If anyone does have a copy of the program, could they please forward it to me at the normal Techno-Info address. CDU, by the time this is published, may have already started such a section as you mention - called the CDU Readers' Page (though at £5 for 20 words I seriously doubt it!). And I thought that that was what this section is! I hope someone can assist.

## DISK GONE WALKIES

Dear CDU,
Please could you ask your readers if any of them have got the V4.1 disk for the EXPERT cartridge for sale as my

own copy of the disk as gone for walkies. Please help.
**M.S.Potten.**

Dear Mr.Potten,
Have any of the readers got a copy of the V4.1 disk for the EXPERT cartridge for sale? There you go, I have done it for you - if any of you out there in Readerland do have the said disk for sale then please write to CDU Techno-Info giving any necessary details, such as how much cash will persuade you to part with it!

## DOUBLE DISAPPOINTMENT

Dear CDU,
I have heard so much in your magazine about the colour printer from STAR, the LC10C that offers Commodore compatibility. I decided to visit the local computer shop but they said that they had never stocked them and so I telephoned a number of other companies who all referred me to other places. Nothing came of my endless calls. I am also trying very hard to get hold of a 1581 disk drive, either new or second hand. I have read that you have one - could you tell me where you got it from as, like with the printer, I haven't had much luck in finding one yet. Could you also possibly suggest a source of supply for the printer? Or are these pieces of hardware simply not available anymore? With a bit of luck you'll be able to give some assistance.
**Stewart Hall, Exeter.**

Dear Stewart,
I know of somebody who is in the same boat as you regarding printers, and he has discovered that the STAR LC10C is no longer available here. I seem to remember that I had a lot of problems back in August last year when I was buying mine because there weren't a lot of them about. If anyone knows of a company that still stocks them, please get in touch. If I were you, I would hunt around for a printer that has a serial interface already installed because a lot of compatibility problems can arise from parallel interface printers. Regarding the 1581, that is definitely not available here anymore. I think that you can get them in Germany if you fancy a trip abrôad. It is a real shame that this drive never took of in Britain - it just didn't get all the necessary advertising, and people were too sceptical of a 3.5" disk drive for the C64. I am glad that I bought one when I did because I have found it to be an invaluable aid - both faster and having a larger capacity than the 1541 or 1571 compatibles. Sorry to have to disappoint you on both scores.

## FURTHER THAN 255

Dear CDU,
Firstly may I say congratulations for an excellent magazine, although not too easy to obtain here in darkest Africa. I bought Your Commodore for years until it became YC (Yukky Comic). Do those people really work for the same company as you? I believe there is still a major need for a serious, yet light-humoured mag, such as CDU. I am also a PC user but would never part with my trusty 64 (nor would my kids allow me to) because of its amazing sound and graphics on the games. How do they manage to squeeze so much stuff into so little memory? I have written some small BASIC programs in the past and never dared to venture any further. Recently, I decided it was finally time to learn something about the mysterious inner workings of the machine and was spurred on by MEMORY SCANNER from November 1990. I have an Action Replay Mk.V Pro Cartridge and found this works not only just as well but will also print these strange writings. For some reason I felt I had to have a list of decimal to hex addresses so I would know where I was looking. The cartridge does hex/dec conversions (eg PRINT $C000) but I was unable to incorporate this into a BASIC program. I next tried writing a dec/hex BASIC program but the calculations I was doing for the higher numbers allowed me to go and make a pot of tea while it was working! Then lo and behold, along comes Volume 4 Number 4 and "Tip of the Month" was the very thing. I couldn't believe how short the program was and studied it with a view to extending it past 255. Now a programmer or a mathematician I ain't, so please, how do you do it? In Volume 4 Number 2 you published a letter from Colin Sanderson, also from South Africa, who was lamenting the non-availability of the latest software. Although some of us live in the sticks here, there are thousands of 64 users and new software is available. We are however seriously lacking in quantity especially where hardware is concerned compared to the UK. Could you tell me what hard drives are available for the 64. I would also be obliged if you could supply me with Colin's address, so that I can contact him. If you fancy printing my letter on your Techno-Info page, you may shorten it to save space. Once again, thanks for a brilliant publication. Keep up the good work.
**Mike Hammond,** P.O.Box 55, Stanger 4450, R.S.A. - (South Africa to most people!).

Dear Mike,
I must admit that the YC crew do in fact inhabit the same edifice as the CDU gang - it's unbelievable that we're all still sane, really. I'm glad that you found the conversion program that I wrote useful. To go on past 255 it relies on you splitting the number into two sections - what is called the high byte and the low byte. The high byte is found by taking the whole number part of the value of the decimal number divided by 256, and the low byte is found by calculating the remainder of the calculation just performed. For example, let's consider 20000. Dividing by 256 gives 78.125, and 78*256=19968. Subtracting that from 20000 gives 32. Therefore the high byte is 78 (decimal) and the low byte is 32. These can then be transferred to hex and lumped together, high byte followed by low byte. 78 in decimal is the equivalent of $4E (hex) and likewise the hex equivalent of 32 is $20. Therefore 20000 in hex is $4E20. To avoid confusion at a later date if you get into programming machine code, the computer always STORES these the other way around, the low byte followed by the high byte. Anyway, you say that you're not a programmer and so on this issue's disk is a program filed as "FURTHER THAN 255" which will not itself display a list of all the numbers from 0 to 65535

because that would take a lot of ink. Instead it asks you to enter a decimal number and then it converts that into a hex number - the process then repeats. I hope that you find the method useful. To avoid printing Colin's address if he didn't want it broadcast I have published yours instead (if you didn't want yours broadcast then hard luck really!). In this way, anyone out there in Africa (including Colin if he's still an avid reader) can write to you.

## GEOS AND NEOS

Dear CDU,
Before I get down to the business side of my letter I would like to say how disappointed I was that you decided to include a list of numbers instead of the regular letters in the July "Techno-Info" section. As somewhat of an amateur those lists were of no use to me, although I guess some people found them useful - and how many times do we need to see "Exploring the 1541"? CDU used to be a decent serious magazine but recently many less-than serious articles have been creeping their way in. Let's have some light-heartedness, but not silly stuff that isn't really related to computing. And who was the wally that came up with the "Wally" idea? Any visitor to the planet could have cracked that competition within seconds! But enough of my criticisms, I would like you to answer a couple of my queries, if you would be so kind. Firstly, I purchased (a long time ago) the package called "Mouse and Cheese" from NEOS. However, I cannot seem to get the NEOS mouse to work within the GEOS packages that I bought off a friend a few months ago. I have selected the mouse driver and done everything else just as the manuals say but still I cannot get the computer to respond properly. What is going on? Secondly, I am one of the few people in this country that owns a 1581 disk drive. However, it seems to play up now and then and some of my disks are rendered completely useless - the directory won't load, programs won't load, and strange enough of all, programs can't be saved. A "short" format has no effect and the only thing that results in the disk being able to be used again is doing a full format of all eighty tracks. Can you tell me whether this is a standard fault in the drives or whether it is just mine that seems to be erratic in its behaviour. I look forward very much to a reply.
**Kevin Braithwaite, Colchester.**

Dear Kevin,
To begin with, I was in NO WAY associated with the publication of the list of addresses that appeared in place of the regular "Techno-Info" section in the July issue. I also was not impressed by the decision to name the list of numbers in such a way that people would think that I had made the decision to cut the letters out. As I have commented to other people, the decision was taken by the higher powers of the building to which the Editorial staff must bow. I can assure you however that a great number of readers did find those lists extremely useful. Likewise with the disk drive article. We know that it has been published three times in CDU now but each issue there are new readers who have not seen it before. As it is such a popular topic it seemed fair to reprint it after we had had so many requests for it. As to the "Weird and Wonderful World of Wally" cartoon, I can't really see why you detest the idea so much. People always complain that there are insufficient competitions, and as soon as one comes up someone isn't impressed. Perhaps the questions were a bit simple but if you feel that way, why not enter the competition? It is very hard to find an amicable balance between the sublime and the ridiculous. Now to your main queries. The NEOS mouse is not really a true proportional mouse in that it acts more like a joystick. The output from it is different to that given by a standard Commodore 1351 mouse and so GEOS does not recognise the input that it receives. The GEOS driver is not compatible with the NEOS mouse although I seem to remember that a driver was written for it - perhaps you could phone Tim Harris of FSSL on 0386-553153 to find out. Your second problem has nothing whatsoever to do with faults in your drive - in fact it is your fault, I'm afraid to say. Once something has been saved on the 1581, the READY prompt reappears on the screen before the drive has finished its activity. If you then assume that everything is finished and switch off your computer, or even just reset it, then you will find that the disk becomes corrupt on every sector. The drive cannot open a channel to read from or write to the disk and therefore it is rendered useless. What you must make sure of is that you wait for the drive light to go out and for head movement to cease before you do anything else on the computer. If you follow those simple rules then you and your 1581 should live together in perfect harmony.

## GENERAL QUESTIONS

Dear CDU,
Many thanks for the directory and scroll routines that you wrote for me - I think there were others out there with the same problem. There are a million questions I would like to give you to answer but to the point there are a few new problems. Concerning the machine code routines that you wrote - how do I transfer machine code to BASIC? If I got a scanner for my C64 computer and scanned a picture, is there any way of saving that picture to disk or tape and putting it into my own programs? One more question: How long have you been programming? You're brilliant at what you do and thank the other staff at CDU as well. Keep the programs rolling because I would hate to see you all go again. Thanks again.
**Shaun Ore, Birmingham.**

Dear Shaun,
There is no real way of transferring machine code into BASIC because it just isn't worth it. The scroll routine cannot be done sufficiently smoothly in BASIC anyway. If you were to purchase a scanner, then it is possible to save these pictures out and use them in your own programs. The scanner saves pictures out to disk usually in one of several formats. These are in the format expected by certain art packages widely available. The pictures can also be loaded by you if you know what

format they are in - in other words you must find out where the bitmap information is stored, and where the screen and colour memory data has been kept. To answer your last question, I have been programming in BASIC for seven or eight years, and I moved onto machine code with much persuasion from my father (who nevertheless can't understand machine language!) about five years ago. I could never sit down with any manual and plough through it all so I taught myself by experimentation once I had become familiar with the command words and, for machine code, the instruction set. I am pleased that you find CDU such a worthwhile magazine and I hope that you continue to enjoy it.

## LASER PRINTING

Dear CDU,
Could you explain the equipment and cables you would require to interface a Commodore 64 to a laser printer? I have GEOS 2.0 and on page 227 of the manual they briefly mention how to do it, but could you give me more details? Would I simply obtain an RS-232C interface and plug one end into the 64 and the other in the printer? Or are other things required? I am thinking of leasing a printer and would like to know how easy it is to connect things together. At the moment I have an electronic typewriter connected to the 64 and this was a question of obtaining a special cable and a special Silver Reed interface, so this was very easy to do. Hoping the interfacing of a laser printer will be the same. The GEOS manual mentions PostScript language - what is this?
**N.K.Taylor, Bournemouth.**

Dear Mr.Taylor,
First of all, PostScript language: It is very difficult to explain what this is in words, but I shall try anyway. The printer, in order to create the fonts and different styles/modes of printing, must have a language installed. The language of some printers is called PostScript and it can only be used in conjunction with programs that use the PostScript language as a means of communicating with the printer. If the program is not PostScript compatible and the printer has the PostScript language then you can't use them together. That, basically, is the idea. It is not over difficult to connect a laser printer to the 64 and to get it to work with GeoPublish or similar. But the range of printers available is large and the interface that you would require would be different for different printers. It would be best if you find perhaps three laser printers that you are interested in, and then telephone the UK Distributors of GEOS, FSSL, on 0386-553153. When the models of printer are known, they should be able to guide you further and recommend the leads and interfaces that would work.

## MAGIC PICTURES

Dear CDU,
I have a bit of a problem that maybe you or one of your readers could help me with. I am trying to load in a "Paint Magic" picture file, then display the picture on the screen. The problem is I can't seem to get the colour right. You see, I am writing a utility program and you can load in a picture file from various art programs. The picture is then displayed and you can do various things which I will not go into just now. I have managed to display Vidcom, Artist 64, Koala, etc., it's just the Paint Magic files that I can't display properly. I converted a file from Vidcom and various others to a Paint Magic file using the Expert picture formatter - at first I thought that the formatter had a bug in it, but I got hold of a file done on a different formatter and both files are the same layout. Most files are about 10K long with 8K for the bitmap, 1K for the screen colours and 1K for the colour memory colours ($D800), but a Paint Magic file is shorter. At the start there is a machine code program to display the picture. If the file is loaded into $0801 then run from $0815, the picture will be displayed, minus the proper colours at $D800 onwards. The displayer program at the beginning of the file takes one byte and puts it in the whole of the colour memory which seems to be wrong for a multicolour picture. The colour memory part of the file seems to be missing. Well that is my problem - it's probably something simple but I can't fathom it out.
**Maurice Le-Vallois, Paisley.**

Dear Maurice,
This is probably due to the way that Paint Magic pictures are created. Although they are multicolour, it can be that the same colour bytes are used throughout the entire picture. The program "The Image System" uses a similar method where ALL the screen memory bytes AND the colour memory bytes are the same universally over the whole screen. This allows only three colours to be used, but it is still, technically, multicolour. Files that are created with a different art package that allows each byte to be different just cannot be converted to Paint Magic files. To view the latter files you should use exactly the same format as the picture loader and store one byte throughout the entire colour memory map - that will be correct.

## MORE TAPE TO DISK

Dear CDU,
Could you please let me know if it is possible to transfer the games I have on tapes to disk on my C128.
**D.A.C.Street, Ottery St Mary.**

Dear Mr.Street,
If the games run in the native mode of the C128 and not in the C64 mode then, if the software is protected and commercially purchased, there is no way it can be transferred easily to disk. If alternatively it runs in C64 mode then you need to buy a cartridge that will enable you to "freeze" the game and copy it out to disk. However, you should note that the use of this sort of device even for making archival backups has now been banned and is illegal. Therefore I'm not permitted to give

you any more details on how to go about it all. These cartridges are useful for many other things as well though and the purchase of one will help with programming and so on. The Action Replay cartridge from Datel Electronics of Stoke-On-Trent or similar will provide you with extra BASIC commands which are very useful. They also have routines that speed up the disk access so that your disk programs will load in a fraction of the time. These sort of items are very useful for improving general capabilities of your machine whilst at the same time providing less savoury features.

## NOT ALLOWED

Dear CDU,
I am thinking of buying either the Action Replay VI or the Trilogic Expert Cartridge for use in transferring games from tape to disk. In your May edition of CDU you had a letter from John Kopsidas of Greece. In relation, I have three queries which I hope you can help to solve. Firstly, which cartridge caters for multi-level games? Next, could you give me some detailed information on each cartridge, including their capabilities and which one would satisfy my needs? Lastly, how much will each cartridge cost? I hope you can help me.
Mark Ward, Plymouth.

Dear Mark,
As I have just said to Mr.Street in the previous reply, the use of copying devices is now banned. Not only the use, but also the manufacture, distribution, import, export and so forth of any device solely for copying is illegal - you are even guilty if you have the parts that could be put together to manufacture some form of copying device. Cartridges and certain pieces of software get away with it because they have so many other functions and therefore they are not defined as having "the sole purpose of copying any other copyrighted product". But it remains that even the most innocent of copying - from transferring from tape to disk, from making a backup copy "just in case" is now illegal and you are not then permitted to do it. I cannot even recommend to you what you should buy in order to do what you want without breaking the law myself. Both cartridges that you mention are around the same price (roughly £30-£40) and both have a variety of useful functions. I myself own AR6 which may cater for some of your needs but the Expert, although not having all the system stuff "on board", may cope better with some of your more specific needs - it depends what they are! Sorry that I can't come straight out and say what you should buy. Just remember if you get found out you could be at the wrong end of a rather hefty fine.

## PARALLEL TO SERIAL

Dear CDU,
Help! Help! There's a lady in distress (well more of a mature dame)! In an effort to upgrade my 128D set-up, I part exchanged my HR5 printer for a Citizen 120D (no manual) which I believed could be used with my computer. I realised I would need an interface and bought a Sprint 128 Centronics to Serial unit from Datel. Also a MkVI AR cartridge plus a Centronics to Parallel lead. Using either interface the print head moves into position and pressing the line feed button moves the paper up a space. However, entering a simple program and using the OPEN1,4,2 etc command produces no response from the printer. Using OPEN1,5 etc has no effect. Using a simple utility to print a directory in conjunction with the C to P lead brings a "Device not present" response. Please, what is wrong - printer, interface or me? One other query, please - I bought some used disks, one of which, Superscript, loads automatically. However Superdesk128 (40 columns) consists of SEQ files which I cannot load. I tried OPEN 1,8,4,"Superdesk,SEQ,R" from a manual with no luck. Should I throw everything away and take up knitting? Or can you save me from a horrible future? Yours in despair.
**Ann Pickston, Manchester.**

Dear Ann,
"What is wrong?" I'll give you a clue - it's not the printer and it's probably not you! You can take up knitting if you like, just make sure that I'm the first to receive a woolly jumper with "CDU Techno-Info" knitted into it! Seriously though, the interface that you require is the Super Graphix Jnr interface from FSSL. Their address is FSSL, Masons Ryde, Defford Road, Pershore, Worcs, WR10 1AZ and you should quote the catalogue number, 441, if you order it. This interface is far more univeral than some of the others that are available and should work perfectly. The SEQ files in Superdesk are not, I would think, the actual main program. Perhaps the disk should boot automatically and then have a special loader to read in these files. In themselves, from BASIC, you cannot load SEQ files to form a program that is runnable. However, I am not familiar with the package but would lay bets on the fact that this disk either should autoboot, or has had the important loader file corrupted or removed.

## PLUS/4 DRIVER

Dear CDU,
I have a problem with Steve Carrie's 65XX program "Plus/4 Printer Driver" from Volume 4 Issue 1 of CDU magazine. I assembled the program using the Plus/4 Assembler from YC magazine. After starting the program using SYS DEC("7000") use of the OPEN command (eg OPEN 1,4 or OPEN 4,4) returns "Press Play & Record on Tape" message. I have checked my input program and it seems OK, so I am seeking your assistance in getting the program to behave correctly. I have listed a Disassembly (using Plus/4 Monitor) in the hope that you will be able to see the error that I am unable to find. In anticipation of your help

**R.J.Patterson, Victoria, Australia.**

Dear Mr.Patterson,
As I do not have a Plus/4 myself I have not been able to try out the program though we have not had any queries from other readers. I have studied your assembly language listing and have found a couple of things that, to me, seem a little odd. They occur at $7005 and $7023. The machine code value for a STX command is normally $8E. At the two locations mentioned the code is $18 and $BE, although the correct command is listed on the right. One of these errors is with the changing of the OPEN vector so this could be what is wrong. Although the commands are all right, the values stored in memory are incorrect (which seems rather odd to me - that you should get the right commands even though the values are incorrect). You should ensure that at the two locations mentioned the values are $8E even if this means assembling the code and then changing the values manually. Do that and then try it out again - with a bit of luck everything should be all right.

## PRINTER HASSLES

Dear CDU,
I own a Commodore 64 and I've just bought a second-hand Commodore MPS801 dot matrix printer with no manual. The printer works well with listings of programs on disks but after RUN is executed, I don't know what command sends results of programs to the printer for printout. Can you enlighten me on what command, if any, is used to send results of written programs after RUN to the printer for a hard copy printout please. I would be grateful if a solution to the above problem can be given to ease my backlog of work schedule.
**Richard Viatonu, London.**

Dear Richard,
I think what you require are the OPEN, PRINT#, and CLOSE commands although I'm not entirely satisfied that I understand your query. To use the printer from within a program, the program must have something like an OPEN 1,4 or OPEN 4,4 command in it. Then PRINT#1,"text" or similar must be used for output to go to the printer. CLOSE is used after everything. If these are not in a program then it won't output to a printer. You could try OPEN 1,4: CMD 1 before you RUN it, and then everything that is PRINTed to the screen will go to the printer instead. To get a hard copy of the listing of a program, enter OPEN 1,4: CMD 1: LIST. I hope that somewhere there is a solution to your problems.

## RELOCATORS

Dear CDU,
For several weeks I have been trying to write a code relocation routine without success. I was hoping you could include one on a CDU disk that I can use. I have several programs and routines that reside at $C000 and I need them moved to other parts of memory. I hope you can help with this problem as I'm sure lots of other people have trouble with this too. I have a basic understanding of machine code and its various techniques if that is of any help. PLEASE HELP! By the way, Techno-Info is my favourite part of the mag (besides the disk, of course).
**Craig Dickson, Solihull.**

Dear Craig,
There was a code relocator published by CDU back in the early days of volume one. One method that you can use is to start at the beginning of your code (where else?) and have a table that contains the "length" of each machine language instruction. For example, LDA $C100 needs three bytes. You read in the first byte, and compare it with different values to see whether it is one like LDA $xxxx, STA $xxxx, ORA $xxxx and so on. Of course you would check for numbers rather than the words. If it is one of those that references a two byte address then you read in the next two bytes and form from them an address. If this address lies between a certain range (that that the code occupies usually) then it needs changing. You calculate the new address using an offset. For example, if you were transferring from 49152 to 24576, each two byte reference would need reducing by (49152-24576)=24576 by coincidence. You will have taken from your table the number of bytes that the command occupies, so simply add this to your counter or something and you will get the start address in memory of the next command. Then just repeat the process until all is done. When the code is completely changed, it is then a simple matter of a FOR...NEXT loop to transfer the code to the new address. Of course that implies that you have programmed this relocator in BASIC. You can do, it will be a great deal easier. If you have any more problems, and if you have a copy, look at some of my programs that have appeared in the past like Screen Slider and Multi-Sprite. These both, I think!, had routines built into the BASIC demonstrations that gave the option of relocating the code. Of course the bytes that needed altering where known, but searching for them is not really a problem.

## RIGHT TO REPLY

Dear CDU,
I would like to respond to "An Odd Bug" printed in the Techno-Info section on page 29 of June 1991's CDU. Mark Fletcher states that the bug manifests itself only in his games software. Could he have a problem with his joystick or associated circuitry as this seems to be the 'X' factor in his diagnosis? I realise this is related to input/output but it is an item of hardware needed for games. By the way, it's a great magazine - The Bible of 64 disk programming! Also, any information available on machine code tape loading without the use of the KERNAL related LOAD commands would be appreciated.
**N.Whittaker, Blackpool.**

Dear Mr.Whittaker,
Thank you for your suggestion. However, I have since discovered that perhaps the PLA Chip of Mr.Fletcher's computer is faulty. This causes the computer to write information to different addresses when it should concentrate on just writing to one address. I would like to thank a different Mr.Fletcher for giving me that information and I would like to thank you, Mr.Whittaker, for the joystick suggestion. I am glad you like the magazine but unfortunately loading without the use of the KERNAL is a very specialised aspect which would take up a large amount of room here. It relies on an in-depth knowledge of machine language and data transfer between tape and computer. Although I do know quite a bit about it and have programmed a tape turbo routine that allows another program to run whilst the main program is loading, I couldn't really give you any information because it was a long time ago and to be honest I cannot remember the finer points. It certainly is difficult and should only be attempted by someone who is very competent with programming.

## SIMON'S ADDRESS

Dear CDU,
First of all, many happy returns for June-the-whenever. Second, please don't mis-spell my name again! Third, find enclosed a page of memory locations for use as a Techno-Info tip. Okay, that's the first few out of the way. If and when you print the tip please print the above address which I neglected to put in the article for my "Hires Converter" program because I moved house soon after submitting it, and wasn't sure where I would be moving to (at least as far as I recall that was the reason, but anyway, the address got omitted - please print it at some time). Thanks also for the information on the CBM1581 you sent me, most helpful. Keep up the good work.
**Simon COLLIS, North Humberside.**

Dear Simon,
First of all, thanks for the birthday greetings - it was June the first (just so that you know next time!). Second, sorry for spelling your name as Simon COLINS a while back - it just isn't possible sometimes to decipher people's signatures if they don't print their name elsewhere. Yes, the mistake is your fault!! Third, thankyou for the tip. Unfortunately, due to that list of "useful information", being published in July (under the conspicuous heading of "Techno Info" when I had nothing to do with it and knew little about it), I shall have to save your tip for a few months because the memory locations were all published in that list. Now to your address for all your loyal fans. It is: Simon Collis, Blacktoft House, Blacktoft, Howden, North Humberside, DN14 7XX. I hope I got that right? Do you want your phone number published as well? I'll leave that until I have your permission of course. The reason for you moving, I recall, was that you left University around that time. How do I know that you ask??? The address on one of your past letters was Lupton Flats, Leeds., which I believe are the University flats. Or didn't you want people knowing? By the way, for everyone else, the information I sent Simon about the 1581 disk drive is no longer available because the drive isn't (at least not in Britain).

## SPELLING MISSTAKES!

Dear CDU,
Sorry to have to groan at you, but I am becoming increasingly aware of silly little spelling mistakes in the text of CDU magazine. Isn't it possible to eradicate these infuriating mistakes. I have noticed that Techno-Info is also suffering from this disease. Hoping you can make my enjoyment of the mag complete.
**Alfred Fox, Liverpool.**

Dear Alfred,
It really is not possible to eliminate the odd spelling mistakes that occur now and then in the text. This section alone is usually over six thousands words long (and this time heading on for 12000! - Count them if you don't believe me) and I do my best to read through it and correct mistakes that occur on the first "pass" of writing. If you like, I could send you my word-processor text file each month and you could correct it for me, but I would guess that you have better things to do than to sit for hours on end checking my spelling. I am sorry if Techno-Info is being lulled into a chasm of inevitable and horrific spelling blunders, but we at CDU do try to kill the bugs. Each article, you see, is spell checked on TWO different occasions on TWO different systems. Once usually on an Amiga, and then again when it is converted over onto the MAC. Incidentally, you may like to know that "eradicate" does in fact only have the one 'r' after the 'e'. You see, wee karnt awl bee purrphekt now, kann wee?

## SUPERSCRIPT BUG

Dear CDU,
I have several problems which I hope you can help solve. I have a C64, 1541 disk drive and a STAR NL10 printer. I can use all the facilities of the printer from BASIC but when using the Superscript word processor, I cannot use some options. Micropro said that it was impossible to use the International Alphabet from within Superscript but, after hours of experimenting, I can now do so. They also said there was no way to produce graphics. However, I have entered the graphics character code (145) into the secondary features of the defaults program and I can produce the graphics on the right-hand side of the Commodore's keys by using the SHIFT key. However, I can find no way of getting the left-hand graphics (using the Commodore key). Can you please tell me if, and how, it can be done? Secondly, the EVEN offset in Superscript for left-hand or even-numbered pages works the wrong way around - it offsets the odd-numbered pages

to the left! Is there a way of correcting this? I recently bought a second-hand C128 and a SFD 1001 disk drive. Unfortunately there was no instruction book for the 1001. According to the C128 System Guide, either one or two CP/M disks were supplied with the original package. I do not have these. Could you please tell me how I can obtain the instruction book for the 1001 and the one/two CP/M disks. I also have Superscript128 and Superbase128, both disks having a label with the number 8050. I presume this refers to a disk drive. Obviously my Superscript 64 disks are not inter-changeable with the 1001 (or 8050?) disk drive. Is there a way to transfer or copy my Superscript 64 data disks to the 1001 (8050?) format to be used via Superscript128 without having to re-type them all again? I am very pleased to see that CDU magazine is still continuing and wish you every success in the future.
**R.Dunley, Chester.**

Dear Mr.Dunley,
You are right regarding the method for obtaining the graphics set, but I can assure you quite confidently that there is no simple way of obtaining the graphics shown on the left hand side of the keyboard. This is because it is impossible to type them in from the keyboard in the main program. The only way to do it would be to hack into the program and work out how it converts the keyboard input to a character on the screen. That is very difficult and I have had a look for you, and I don't think that it would be a simple matter of changing the odd byte. Besides, altering the program would be a bit naughty. Regarding the EVEN OFFSET function, I had never noticed the fault until you mentioned it. Again, I can see no simple way of correcting the bug without having a detailed knowledge of the workings of the machine language program. It may be that newer versions work. Sorry that I can't help you there. When someone recently asked for the CP/M disks we had a great response and many people wrote saying that they would be happy to supply the two. If those people would be so kind as to write again, then I shall be only too pleased to pass the disks on to you. With respect to the disks, there is no easy method of transferring the information. The only thing that I could think of which may or may not work, is to connect the 1001 to the 128 at the same time as the 1541 if possible (and I can't see why it shouldn't be). Then write a BASIC program that reads in the files from the 1001 with the appropriate device number, and then writes them back out via a different channel to the 1541 which will be a different device number. If that does not work then I can see no way of doing it. Sorry that I can't be a bit more
positive with replies to your questions.

## USER PORT CHANGES

Dear CDU,
I have owned and used a C64 for many years and since buying a second hand disk drive last year I have bought several issues of CDU. However, I have not been able to print out from any of the programs with their print options as I run a Centronics type printer in the user port. The business type software that I use have the options of different types of printer. I particularly found the Cheque Book Organiser useful and have altered the program for use on Centronics type printers plugged into the user port without an interface. Would you like a copy of the disk or the alterations that can be made? I am now going to try it on another program.
**R.G.Johnson, North Yorkshire.**

Dear Mr.Johnson,
Thankyou very much for your kind offer. If you would send me a list of the alterations that are necessary to the Cheque Book program and any others that you have discovered to allow them to work with a printer as device two, through the user port, then I shall be only to pleased to send them on to any other readers that request them, now that they know that they can get hold of them.

## WHAT IF IT'S LAST?

Dear CDU,
I have written a BASIC program which allows me to save, on a weekly basis, information regarding our business. This data is saved in SEQUENTIAL files, with the filenames prefixed with the relevant week number (01 to 52). In order to update the file, at the moment it is necessary to type-in the week number of the preceding week in order to read that data into the computer. What I am looking for is a routine which I can incorporate into my main program which, when that program is run, will automatically read into the computer the last sequential file saved, irrespective of its week number. In other words, in much the same way as LOAD"*",8 will load the FIRST file on a disk, what I require is a routine which, from the main program, will read into the computer the LAST file on my data disk. (C64, 1541-II set-up). Can you help???
**W.J.Wilson, Sutherland.**

Dear Mr.Wilson,
A tricky one this, and a rather cunning solution. There is no stand-alone command that loads the last file in but there is a method that you can use. You write a very short SEQ file to the disk called, for example, "LAST". To this file you simply write the name of the file that has just been saved by the program. Each time the program is RUN, you can load the file called "LAST", read in a string, and then use this as the filename to load. For example, when saving, do something like:

```
OPEN 3,8,3,"@0:"+WK$+"FILE,S,W" where WK$ is the
week in the form 01 to 52
PRINT#3,....
CLOSE 3
```

then follow that with:

```
OPEN 3,8,3,"@0:LAST,S,W"
PRINT#3,WK$+"FILE"
CLOSE 3
```

Then whenever you wish to know what the name of the last file stored was, irrespective of its week number, you

simply do the following:

**OPEN 3,8,0,"LAST,S,R"**
**INPUT#3,A$**
**CLOSE 3**

The variable A$ will contain the file that you require so just use that when OPENing the next file. With a bit of luck you will be able to incorporate something like that into your main program without too much trouble.

## 40 NOT FOR ME

Dear CDU,
I have been using my Commodore 64 since 1982. There is no doubt that this is a wonderful machine. I have used this for programming, for games, and also for letter writing and so on. The only thing which I did not like is its 40 column screen. I agree there is software available which can make it 40 to 120 columns but then you have to scroll the screen sideways to view the text which makes it very difficult to see at one glance if there is any mistake. Only for this reason I am thinking of changing to a new computer. Before that, I would like to ask you if it is possible to make it 80 column either through software or through hardware. If so, how?
**S.N.Madahar,**

Dear Mr.Madahar,
It is possible to turn the C64 into an 80 column machine without the use of scrolling and depends on the high resolution graphic screen. Word-processors such as Tasword will allow you to enter text in 80 columns and some programs that are used for amateur radio allow a 106 column display! I can't recommend any programs that simply turn it into 80 column mode and allow you to enter text and programs as normal because I am not aware of any. You never know, one may appear in CDU sooner or later. But don't throw your C64 away just because you don't like the forty column display. If this is the only thing that you are not satisfied with it seems a bit of a shame to get rid of such a powerful machine in preference for one which you may not be able to program as easily but has a built-in 80 column mode. Perhaps you could look for a C128 - that has an 80 column mode and most of the GEOS software available (spreadsheets, databases, etc.) supports that 80 column mode as does a lot of other software. Keep an eye out for an 80-column driver for the C64. Though a bit rare on their own, they are available.

## UPDATE

Many thanks to everyone who wished me happy birthday and also to those of you that sent me cards - especially MR ERIC FROST who virtually hit the nail right on the head where anticipating the day of delivery was concerned! All much appreciated. This UPDATE, I am going to give you a bit of news that I have received in the form of a Press Release:

MICROSNIPS Mail Order have now moved to new larger premises in Birkenhead. The new address is 25-29 Grange Road West, Birkenhead. And the Mail Order phone number is 051-650-0500, fax 051-650-0506. Microsnips not only serve the UK with computers, software, accessories and business equipment, but also export to Europe, Canada, USA, Australia and Israel to name but a few and of course all British Forces Personnel worldwide.

## WHERE TO WRITE

If you are experiencing any computer-related problem, or you simply wish to air your views or have a tip published, then please write to me, Jason Finch, at the usual address:

**CDU Techno-Info**
**11 Cook Close**
**Brownsover**
**Rugby**
**Warwickshire**
**CV21 1NG**

Please do not send your letters to the CDU office as this can result in a delay in you receiving a reply or having your letter published. Thanks - see you all again next time.

## TIP OF THE MONTH

This month we have two tips to let out of the bag. Both of them come in the form of programs which you will find on this issue's disk! - one is for the 64 and the other for the 128. The first one is from MARK WRIGHT OF ESSEX and concerns the RASTERBARS.BAS program, and the second is from DAVID HENRY and allows you to use the numeric keypad of the C128 for easy entering of DATA statements. It converts the '+' into a DELETE key, the '-' into the word DATA, and the '.' into the comma. Thank you both for the programs. Here are the letters/explanations:

Dear Tip of the Month,
I thought I would send this little idea in as I know a lot of people out there are beginners in programming. So to liven up their programs I have added a few extra lines to RASTERBARS.BAS to enhance the output from it. All it does is moves the bars created with RASTER MASTER up and down the screen at whatever speed is required. Not the best of tips but I have found it handy and I hope other people will. I have enclosed a listing of the program and am hoping it will get published. Keep up the excellent work and try and add a game or two to each disk to liven

it up for the beginners.
**Mark Wright.**

Dear Mark,
You would not believe the number of people that write asking for the odd game now and then - personally I prefer it as well. However, you would also not believe the number of people that write complaining when even one game is put on a CDU disk. It is very hard to find a balance. Once again thanks for your tip.

Dear Tip of the Month,
This program allows BASIC DATA statements to be entered much quicker by altering the numeric keypad. It is best used in conjunction with automatic line numbering. After starting the routine the keys are as detailed above. The '+', '-' and '.' keys on the QWERTY keyboard are not affected and work as normal. The source code was written using 6510+ Assembler.

**Dear David,**
Thankyou for providing the readers with another labour-saving routine. To use it, it will be necessary for you to load and run the program in C128 mode. The changes will then be made and it will be ready for use. Thanks again, David.

## THE TECHNO-INFO CHALLENGE!

The time has come for another Challenge. This is the third of the Techno-Info Challenges, the next one coming in the December issue! First of all we must deal with the winners from last time when I asked you to design and write a Tic-Tac-Toe program that simply couldn't be beaten. We had fewer entries for this than we did for the Prime Number Challenge and the winner receives fifteen pounds worth of computer gear of his choice absolutely FREE. The overall winner was PAUL GANDER OF GOSPORT who should have received the items of his choice by now. His program was a mere SIXTEEN lines long and not only included a numbered grid but also a lot of text and "user-friendly" aspects. His winning entry is filed on this month's disk for you all to peruse. And yes you've spotted it - Paul also won the first Challenge. Come on you lot out there, someone has to be able to better this master! The top five entries were received from the following people:

1. with 16 lines - Paul Gander of Gosport.
2. with 18 lines - Murray Janell of Hanstedt, Germany.
3. with 19 lines - Mark Carroll of St.Austell.
4. with 26 lines - Peter Weighill of Bourne.
5. with 31 lines - Roger Harris of Birmingham.

## THIS MONTH'S CHALLENGE

**This month I have written a program for you so the first bit of good news is that you don't have to do the programming. The only problem is my own programming is becoming a bit rusty (not really!) and I have made a number of errors in the program which means that it doesn't do exactly what it should, and most of the time it stops with an error even before the title screen is complete. There are a few obvious errors that result in the program being aborted but most of them must be subtle little ones that I haven't been able to see! The program is 94 lines long and a little birdie tells me that there are TWENTY lines that contain errors. Your task then, this time, is to find out which of the lines are errornous and to correct the details. Remember, they are not only blatant programming blunders - two in particular are very subtle methinks. Of course, you can't rectify the program if you don't know what it should do, so I shall briefly explain what I originally intended the program, which is a simple game, to be like.**

**It opens with a wonderful title screen welcoming you to this Challenge. The program is a bat and ball game which uses a joystick in port two as its means of control. You are supposed to press fire to start the game. The screen goes black and then the following things are supposed to happen. The information lines at the top are scrolled onto the display and then a neat rectangular border is drawn around the main play area of the screen. This should go along the bottom line and also near the top of the screen leaving a blank line between this border of character squares and the text that scrolls in from the right. Of course two sections are on the left and right as well to complete the border. Jewels and obstacle blocks (which are sort of hatched squares) are plotted randomly within this area and a check is performed to make sure that there is a clear space around each character. You start with 15 jewels on the screen and there should be twice as many obstacles as jewels to start with. Two bats are on the screen - one at the top and one at the bottom and you move these left and right together to stop the ball hitting the upper and lower boundaries. If the ball hits a block you get 1 point and the ball rebounds vertically. If the ball hits a jewel you are awarded 10 points. If you don't manage to knock the ball with the bats to keep it in the play area, then you lose a life and the screen is redrawn again randomly. You only have to collect as many jewels as were left when you lost your life but there should still be 30 obstacle squares. When you have lost three lives the game is over and you must start again. That, anyway, is the theory behind what is supposed to happen. You should note that I intended the bat not to be able to go right into the corners of the screen and so it is not always possible to get the ball if it bounces into the corners of the screen. This need not be rectified because it is intentional.**

## THE RULES

**So you must find all of the TWENTY errors and deliberate mistakes and change them to what they should be in order for the game to function as described above. No lines are to be added and none are to be removed. Most mistakes are mathematical symbols, variable names or numbers that have been typed incorrectly and there is the occasional misspelt word. Only one of the lines requires a large change - but that is still just altering numbers so that they are the right way around. You are not permitted to alter**

**routines and simple aesthetic changes that better match your preferences are not counted. When you think you have found as many mistakes as you can, send in your corrected version, either printed out or on a disk. If anyone manages to spot the TWENTY deliberate errors then the first such entry out of the hat will receive the prize. You are going to have to rely on a bit of luck also this time because it would be unfair to say that the first correct entry received is the winner as everyone receives their copies at different times. Remember, there may be more than 20 things wrong - it is 20 LINES that are incorrect, not 20 CHARACTERS/LETTERS.**

**I hope you enjoy this Challenge, the closing date being 30th September. If you live outside the British Isles then the closing date will be extended to around the end of the year due to the fact that you don't receive your magazine until later. Winners will therefore not be announced in the next Challenge section, they will be listed in the March 1992 issue. I wish everyone the best of luck with this, probably the toughest Challenge yet. Send your entries to the address below.**

## WHERE TO WRITE

If you have any computer-related queries, or if you would like to send a tip or a Challenge entry, then you should send ALL necessary details to the following address (enclose a stamped addressed envelope if you would like a personal reply before publication of your letter):

**CDU Techno-Info**
**11 Cook Close**
**Brownsover**
**Rugby**
**Warwickshire**
**CV21 1NG**

Keep your letters coming and let us know what your views on various computing matters are as well. I shall join you again in the October issue.

**PLEASE NOTE:**
**As from the December 1991 issue of CDU the FULL ADDRESS will be printed for letters that are published in this section. We have had so many people wishing to converse with other readers through this section that this is the best way. If you DO NOT wish your full postal address to be published then please attach a footnote to your letter to that effect. I thank you very much and hope that this small change will improve inter-reader communication even further - Regards... Jason.**

# RETURNER

**Return to the menu at any time with this ingenious program**

The problem of keeping records of what's on your disk was addressed in COMMODORE DISK USER Vol 1, No 2 (Jan/Feb 88) by MENU MAKER. This useful utility allowed you to load any program direct from the Menu, but to return to the MENU you had to add the load statement to each program on the file, which is easy enough to do, but usually means putting check questions in and having to end the running of the program. you could also do it by pressing Run Stop and reloading "Menu", but that's rather a waste of time.

The program RETURNER allows you to leave a program at any point or time, and return to the menu to select an alternative program without all the problems outlined above.

## LETS RETURN

On loading and activating the "RETURNER" program code, the first job that it does is to set up the "RESTORE" key, and automatically load a MENU.

Obviously, for the program to work correctly, you must have a "MENU" program on your disk. We publish one every month. Alternatively, you could write one of your own. To get the best results, it is advisable to have

at least two, if not more, programs on your menu.

Once you have your menu program running, you select which program you want as normal. On running the selected program, you can press the "RESTORE" key at any time to return to your MENU program.

## IF THE RETURNER WON'T WORK

There are two reasons why Returner may not work:

1 The program in memory is using the NMI interrupt, or it is restoring the interrupt vectors to normal values. You need to alter your program (if possible) so that it's not using the NMI interrupt.
N.B. The NM1 interrupt values are located at 792-793 ($318-$319).

2 The memory location where "Returner" is located is being used by the program being run. See solution below.

## PROGRAM CHECKER

This program will provide you with alternative memory locations to hold the "Returner" program. Load "CHECKER",8 & run. This will give the following instructions on screen.

First Type in N
Then Load Menu, Load A File, & Reset
Load Menu, Load Another File, Reset
Repeat Until all Files Loaded
Load Checker & Type Y
Have You Wiped Before?

When running, "Checker" will show a blank screen until processing is complete. This will take approximately three minutes and then show on the screen:

Ready
**Load"Menu",8**

Press Return. Run the menu, then load first file, then reset computer by pressing RUNSTOP & RESTORE (or by using a Reset cartridge) DO NOT SWITCH THE COMPUTER OFF & ON.

Reload the menu, run the next program, then reset as before and continue until all files have been loaded. Once all programs on the disk have been run, reload CHECKER, run, and type in Y to the question.

Option is given to output to printer as well as screen. The result of the program is a list of free memory positions available for insertion of the program Returner. You can select the desired position to place Returner (if no space available it will tell you so, i.e. Returner cannot be accommodated on that disk).

To change the position of returner in memory

**Load "EDIT",8 and run**

The screen will ask for the new start address. You can enter it in Hex or Decimal numbers (the Checker program provides location values in decimal). Once entered, press return.

The screen will then ask you to state which file you want to be automatically loaded when you press the RESTORE key. If not MENU, then alter to your own requirements - press Return.

The screen will then ask what name you wish to give the Returner program. Put your own name in. The screen will then ask "Are you sure?" - if not, type N, if you are, type Y. N returns you to the beginning of the screen, and you will have to re-input memory location etc. When you type y, it will save the newly-named program to disk.

## STARTING UP YOUR COMPUTER

When you start up your computer, you initially load and run the program you named and saved above. It will automatically install the RESTORE key function and run the program you asked to be first loaded (see above). I recommend that it is the menu as pressing the RESTORE key will automatically return you to this program until the computer is switched off.

**NOTE: A good place to put the "RESTORER" program in memory is 820 ($334), as it is not normally used when using disk. It relates to the tape drive use, so you won't be able to use the tape drive.**

If you have a multipart game, Restorer can be used to restart the game by asking it to automatically load in the first part of the game as part of its program (see above). The start program must, however, have a Basic line number at the beginning e.g. 10 SYS3000.

## ADDITION TO RETURNER INSTRUCTIONS

It is not recommended to press RESTORE during the use of the disk drive or printer, as the menu may not work properly if you do so.

If you need to press RESTORE while the printer or disk is in use and the menu doesn't work properly, then just press RESTORE again and the menu should work properly.

...it's dynamite!

# POWER CARTRIDGE

FOR YOUR COMMODORE 64/128

"...unbelievable value for money" ZZAPP! Dec 89

SO MUCH FOR SO LITTLE

- POWER TOOLKIT
- POWER MONITOR
- TAPE & DISK TURBO
- PRINTERTOOL
- POWER RESET
- TOTAL BACKUP

..."Money well spent" YC/CDU Jan 90

42 Pg Manual - "Damned Good Handbook" CCI Jan 90

AVAILABLE FROM ALL GOOD COMPUTER RETAILERS

TRIED AND TESTED - OVER 100,000 SOLD IN EUROPE


...."highly recommended for C64 users" CCI Jan 90

YOU WILL WONDER HOW YOU EVER MANAGED WITHOUT IT


## POWER TOOLKIT

A powerful BASIC-Toolkit (Additional helpful commands) that considerably simplifies programming and debugging.

| | | |
|---|---|---|
| AUTO | HARDCAT | RENUMBER |
| AUDIO | HARDCOPY | REPEAT |
| COLOR | HEX$ | SAFE |
| DEEK | INFO | TRACE |
| DELETE | KEY | UNNEW |
| DOKE | PAUSE | QUIT |
| DUMP | PLIST | MONITOR |
| FIND | ILOAD | BLOAD |

RENUMBER : *Also modifies all the GOTO's GOSUB's etc. Allows part of a program to be renumbered or displaced.*
PSET : *Set up of printer type.*
HARDCAT : *Prints out Directory.*

The toolkit commands can be used in your programs.

## DISK TOOL

Using POWER CARTRIDGE you can load up to 6 times faster from disk.
The Disk commands can be used in your own programs.

| | | |
|---|---|---|
| DLOAD | DVERIFY | DIR |
| DSAVE | MERGE | DEVICE |
| DISK | | |

MERGE : *Two BASIC programs can be merged into one.*
DISK : *With DISK you can send commands directly to your disk.*

## TAPE TOOL

Using POWER CARTRIDGE you can work up to 10 times faster with your data recorder. The Tape commands can be used in your own programs.

| | | |
|---|---|---|
| LOAD | SAVE | VERIFY |
| MERGE | AUDIO | |

## POWERMON

A powerful machine language monitor that is readily available and leaves all of your Commodore memory available for programming.
Also works in BASIC-ROM, KERNAL and I/O areas.

| | | |
|---|---|---|
| A ASSEMBLE | I INTERPRET | S SAVE |
| C COMPARE | J JUMP | T TRANSFER |
| D DIS-ASSEMBLE | L LOAD | V VERIFY |
| | M MEMORY | W WALK |
| F FILL | P PRINT | X EXIT |
| G GO | R REGISTER | $ DIRECTORY DOS Commands |
| H HUNT | | |

## PRINTERTOOL

The POWER CARTRIDGE contains a very effective Printer-Interface, that self detects if a printer is connected to the Serial Bus or User-Port.
It will print all Commodore characters on Epson and compatible printers.
The printer-interface has a variety of set-up possibilities. It can produce HARDCOPY of screens not only on Serial printers (MPS801, 802, 803 etc) but also on Centronic printers (EPSON, STAR, CITIZEN, PANASONIC, etc).
The HARDCOPY function automatically distinguishes between HIRES and LORES. Multi-colour graphics are converted into shades of grey. The PSET functions allow you to decide on Large/Small and Normal/Inverse printing.
The printer PSET functions are:

PSET 0 - *Self detection Serial/Centronics.*
PSET 1 - *EPSON mode only.*
PSET 2 - *SMITH-CORONA mode only.*
PSET 3 - *Turns the printing 90 degrees!!*
PSET 4 - *HARDCOPY setting for MPS802/1526.*

PSET B - *Bit-image mode.*
PSET C - *Setting Lower/Upper case and sending Control Codes.*
PSET T - *All characters are printed in an unmodified state.*
PSET U - *Runs a Serial printer and leaves the User-port available.*
PSET Sx - *Sets the Secondary address for HARDCOPY with Serial Bus.*
PSET L1 - *Adds a line-feed, CHR$ (10), after every line.*
PSET L0 - *Switches PSET L1 off.*



## POWER RESET

On the back of the POWER CARTRIDGE there is a Reset Button. Pressing this button makes a SPECIAL MENU appear on the screen.
This function will work with many programmes.

CONTINUE - *Allows you to return to your program.*
BASIC - *Return to BASIC.*
RESET - *Normal RESET.*
TOTAL BACKUP DISK - *Saves the contents of the memory onto a Disk. The program can be reloaded later with BLOAD followed by CONTINUE.*
RESET ALL - *RESET of any program.*
TOTAL BACKUP TAPE - *As BACKUP DISK but to TAPE.*
HARDCOPY - *At any moment, prints out a Hardcopy of the screen. Using CONTINUE afterwards you can return to the program.*
MONITOR - *Takes you into the Machine language Monitor.*

BDL

Bitcon Devices Ltd

88 BEWICK ROAD
GATESHEAD
TYNE AND WEAR
NE8 1RS
ENGLAND

**Tel: 091 490 1975 and 490 1919 Fax 091 490 1918**
**To order: Access/Visa welcome - Cheques or P/O payable to BDL**
**Price: £17.30 incl. VAT.**
**UK orders add £1.20 post/pack total - £18.50 incl. VAT.**
**Europe orders add £2.50. Overseas add £3.50**
**Scandinavian Mail Order and Trade enquiries to: Bhiab Elektronik, Box 216, Norrtalje 76123, SWEDEN. Tel: + 46 176 18425 Fax: 176 18401**
**TRADE AND EXPORT ENQUIRIES WELCOME**

# NEW THE COMPLETE COLOUR SOLUTION

£179

*Vidi ... No 1 in UK & Europe (Leading the way forward)*

**Amiga Computing:** The best Amiga digitiser has had the technicolour treatment. Vidi must be one of the most exciting peripherals you can buy for your Amiga.

**Micro Mart:** When I first saw Vidi "in the flesh" as it were, at the CES show last September it looked to be the answer to a frustrated Digi View owner's dreams – in fact to see pictures appearing on screen without the customary two minutes wait seemed almost too good to be true. I have consistently produced more good quality pictures in the short time I have had Vidi than I ever did with Digiview.

**Zero:** Now under normal circumstances cheap usually means poor quality but this is not the case with Rombo. Why? cos Vidi-Amiga is the best digitiser for under £500 and I've tried them all.

**Amiga Format:** Where quality is concerned, Vidi produces some of the best results I've seen on any digitiser at any price.

**Amiga User International:** The latest addition to the Rombokit is called Vidi-RGB and brings this already impressive package to the realms of totally amazing. CONCLUSION: Who will find Vidi-Amiga useful? The answer to this is almost anyone with a video recorder or camera and a passing interest in graphics.

Get the most out of your Amiga by adding:

## "The Complete Colour Solution"

The Worlds ultimate creative leisure product for your Amiga. Capture dynamic high resolution images into your Amiga in less than one second.

## And Look No Filters

Images can now be grabbed from either colour video camera, home VCR or in fact any still video source. The traditional method of holding three colour filters in front of your video camera is certainly a thing of the past. Because Vidi splits the RGB colours electronically there are no focussing or movement problems experienced by some of our slower competitors. Lighting is also less of an issue as light is not being shut out by lens filters. Put all this together with an already proven Vidi-Amiga/VidiChrome combination and achieve what is probably the most consistant and accurate high quality 4096 colour images ever seen on the Amiga.

The colour solution is fully compatible with all Amiga's from a standard A500 to the ultimate A3000. No additional RAM is required to get up and running.

You will see from independant review comments that we are undoubtedly their first choice and that was before the complete solution was launched. If you have just purchased your Amiga and are not sure what to buy next, then just read the comments or send for full review and demo disk.

*Actual unretouched digitised screenshot*

## Features ...

- Grab mono images from any video source
- Capture colour images from any still video source.
- Digitise up to 16 mono frames on a 1meg Amiga.
- Animate 16 shade images at different speeds.
- Create windows in both mono & colour.
- Cut & Paste areas from one frame to another.
- Hardware and software brightness & contrast control.
- Choice of capture resolutions standard & Dynamic interlace.
- Full Palette control.
- Add text or draw within art package.


ROMBO Limited

***Full colour demonstration disk available for only £1.95 to cover P&P.***

6 Fairbairn Road, Livingston, EH54 6TS. Tel: 0506-414631 Fax: 0506-414634